The Strategy and Realization of Enterprise Integration

A Thesis Submitted for the Degree of

Doctor of Philosophy (by publication)

by

Raymond Wu

in

FACULTY OF INFORMATION TECHNOLOGY

UNIVERSITY OF TECHNOLOGY, SYDNEY

AUSTRALIA

August 2007

© Copyright by Raymond Wu, 2007

Certificate of Authorship and Originality

Date:	August 1, 2007
Author:	Raymond Wu
Title:	The Strategy and Realization of Enterprise Integration
Degree:	Doctor of Philosophy by publication

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in this thesis.

Signature of Candidate

Production Note:
Signature removed prior to publication.

Acknowledgements

I would like to take this opportunity to express my sincerest appreciation to my supervisor, Associate Professor Jie Lu, for her enormous academic guidance in my research and publication. I would also like to thank the Faculty of Information Technology. The opportunity to pursue the PhD degree has allowed me to take a further step toward my career goal through a transformation to a more formal academic research. It has provided me with wider exposure to the mediation between academy and industry as well. In addition, I thank the University of Technology, Sydney for facilitating a quality education to enhance my background in academic publications, and for instruction in how to go about being a scholar.

I would like to express my gratitude to my parents. My mother has continued her moral support for my study; my father, who motivated me repeatedly to pursue my career goal, has been in heaven since 2005, and I am obligated to follow my promise in pursuing the goal.

I extend my appreciation to Professor B. Rugg at New York University, for facilitating a good educational environment and sponsoring fellowship in my early academic life. Thanks also to Professor T. Teng at New York Tech. for providing financial aid for me to complete my M.S. degree in Computer Science.

Many thanks to the companies I have served over the past 24 years - IBM, Oracle, EDS, Vodafone, Citibank, and the US Federal Government. They have provided a solid environment in industry research and broadened my IT background. My co-workers at IBM frequently engage in discussion with me about my publication, and have already taken part of the concept into realization, helping me to serve the IT industry better.

Also many thanks to my colleagues at IBM Lab. in Germany; A. Cameron in Australia; and Otsuki-san in Japan. The friendship of these individuals has lightened my life and kept me moving forward in these 20 and more years of my IT career to date.

Finally I would like to show my appreciation to those journal and conference editors-in-chief and reviewers who not only published my papers but also provided valuable viewpoints and strove with me to make the research papers more readable and to maximize the knowledge contribution to the public arena.

Dedicated to

~ my Parents and the Ones in memory ~

Table of Contents

Table of Contents v

List of Figures and Tables viii

Abstract ix

CHAPTER 1 INTRODUCTION 1

1.1 Personal Background 1

1.1.1 Academic Background 1

1.1.2 Research Background 2

1.1.3 Industry Background 3

1.1.4 Motivation 3

1.2 Problem Identification and Key Contributions 4

1.2.1 Problem Identification 4

1.2.2 Contributions 4

1.3 Inter-chapter Interference and Coherence 7

1.3.1 Paper Structure Aligned to Research Coherence 7

1.3.2 Statistics of Publications 8

1.3.3 Research Flowchart 9

1.4 List of Publications 11

1.4.1 Publications in Academic Format 11

1.4.2 Publications sorted by Category 15

CHAPTER 2 LITERATURE REVIEW 17

2.1 Enterprise Integration Issues 17

2.1.1 Architecture Misalignment 17

2.1.2 Issues in Micro View and Semantic Interoperability 18

2.1.3 Weakness in Common Practices and Industry Framework 19

2.2 Unique Characteristics in Semantic Enterprise Integration 22

2.3 Originality of Knowledge Contribution 23

2.3.1 Cornerstones of Enterprise Integration 23

2.3.2 Concept of Enterprise Vertical Integration 24

2.3.3 Strategy and Foundation for Enterprise Interoperability 24
2.3.4 Synthesis of Fine/Coarse-Grain Enriched Services 25
2.3.5 Scenarios of Business Impact 26

CHAPTER 3 CONCEPTS 27
3.1 The Nature of Integration Ontology 27
3.1.1 Reason of Ontology and how it is managed 30
3.1.2 A Comprehensive Vision of Integration in the Vertical Direction 30
3.1.3 The New-Age Service, Integration, and Dimensions 31
3.1.4 The Big Picture of Ontology, Framework, and Strategy 31
3.2 Preliminary of the Strategies and Approaches 32
3.2.1 Common Elements and Aggregation 32
3.2.2 Component Synthesis and Interoperability 33
3.2.3 Semantics, Metadata, and Mediation 33
3.3 Preliminary of the Framework and Roadmap 34
3.3.1 Coherence of Micro and Macro View 34
3.3.2 Interference Models between Framework, Strategy, and Roadmap 34
3.4 Four Pillars Concept in deriving Framework 34

CHAPTER 4 STRATEGIES 36
4.1 Strategy and Benchmarking for Service-Based Design 36
4.1.1 SCIS and Service Component Design 36
4.1.2 Strategy of Enterprise Alignment 37
4.1.3 Strategy of Semantics Interoperability 37
4.2 Metadata Strategy in Component Mediation and Data Aggregation 39
4.3 META Framework Strategy 39
4.3.1 Multiple Disciplines and the Human Factor 39
4.3.2 Knowledge Base and Intelligence 40
4.3.3 Pattern Benchmarking 40
4.3.4 Service Layer Mediation using Hierarchical Abstract 41
4.3.5 Strategy in Semantics and Mediators 42
4.3.6 Common Component Identification 42
4.3.7 Strategy in Virtualization 44

CHAPTER 5 FRAMEWORK 45
5.1 META Framework Approaches 45
5.2 META Framework Interferences 46
5.3 META Pillar 1 - Method and Validation 47
5.4 META Pillar 2 – Evaluation and Analysis 48
5.5 META Pillar 3 – Techniques and Design Principle 49
5.5.1 META Techniques 49
5.5.2 META Design Principle 50
5.6 META Pillar 4 – Application and Virtualization 50

CHAPTER 6 SUMMARY 54
6.1 Summary of This Thesis 54
6.2 Key Contributions 56
6.3 Limitations 57
6.4 Future Work 57

APPENDIX 59
Appendix 1 Statement on Jointly Authored Contributions 59
Appendix 2 References regarding to My Publications 60
Appendix 3 Bibliography 62
Appendix 4 Publication List 72

PUBLICATIONS 73
Section of Publications in Concept 73
Section of Publications in Strategy 149
Section of Publications in Method 240
Section of Publications in Evaluation 292
Section of Publications in Techniques 356
Section of Publications in Application 439

List of Figures and Tables

List of Figures

Figure 1 Comparing Interactive Model and Influence Model 3

Figure 2 Coherence of Thesis Chapters and Publications 8

Figure 3 Four Pillars of META Framework 20

Figure 4 Origin of Integration Ontology 27

Figure 5 Vertical Coherence in Micro and Macro View 29

Figure 6 Three-Step Approach for Achieving Virtualization 31

Figure 7 Integration Strategy in Vertical and Horizontal Directions 37

Figure 8 Deconstruction and Synthesis in Early Transformation Stage 38

Figure 9 Comparison of Enriched Service Layer and Conventional Design 41

Figure 10 Synchronization between Industry Mainstream and Enterprise Plan 43

Figure 11 Enterprise Coherence in Constructing Framework 45

Figure 12 Strategy and Roadmap in Multi-Discipline Framework 46

List of Tables

Table 1 Publication Statistics 8

Table 2 Paper List (Sorted by Category) 15

Table 3. Examples of Business Scenarios and Contribution 26

Abstract

In recent years we have experienced exponential growth in business innovation, emerging technology, and integration complexity. With this unprecedented growth, the priority of enterprise integration has shifted from patching solutions to the governance of agility. Enterprise integration mainly deals with interoperability between virtual and physical worlds, which is thorny by its very nature. In order to cope with rising complexity, interference coherence between business, service, and physical components is crucial. Instead of consolidation from fragmentation, an iteration approach is taken in driving concept and strategy into realization. The empirical statistics indicate that the anatomy of ontological research is essential for producing an overview of interoperability. The author's numerous research projects demonstrate a number of factors critical in generating higher productivity and lower risk. These factors include a higher visibility of atomic elements, a well-specified service, and a precise architectural alignment.

By taking these successful factors into realization, this thesis proposes enterprise vertical integration, employing a three-step strategy of componentization, transformation, and virtualization. Componentization derives an ontology of atomic elements for the service-based foundation. In transformation, service components are produced from these raw elements, using a multi-discipline and three-dimensional approach to achieve component synthesis. The final step, virtualization, is the objective of enterprise integration. Virtualization establishes the enterprise skeleton and achieves a common-service mainstream in the industry. Experiential evidence indicates that this higher-level, three-step approach works effectively in minimizing risk and increasing productivity. There is particular benefit for projects of higher complexity and larger scale.

Given the incessant business change inherent in our chaotic new age of computing, the three-step approach relies on a new framework to streamline realization and cope with project complexity. A Method, Evaluation, Techniques, and Application (META) framework addresses the interference between virtual and physical layers. In this initial process it develops component validation, analysis processes, and synthesis techniques for service transformation.

It then develops service components and common services for service virtualization. This thesis proposes a four-pillared approach to support the META framework. It also proposes sub-area concepts such as "pattern" and "state" to enhance the capability of the framework before moving it into the industry mainstream.

This thesis distinguishes itself from existing literature in that very few studies in this field address real enterprise-scale integration. None of the reviewed literature copes with the fundamental work of enterprise issues such as ontological research or high-level strategy as proposed by this thesis.

CHAPTER 1

INTRODUCTION

1.1 Personal Background

Born in 1955, I am currently working at IBM APAC Lab., Beijing as leader of Service Oriented Architecture (SOA) Management. I have also worked with IBM (Germany, Singapore, and Canada), Oracle, EDS, Citibank, Vodafone, Lucent, and the US Federal Government over the past 20 years. My roles have included Chief/Enterprise Architect, Chief Technology Officer, Associate Researcher, and IT Director. My geographical exposure has included the Americas, APAC, and Europe. I am a member of IEEE Computer Society and ACM, a guest editor of Inderscience publisher, and guest lecturer for a number of universities/joint programs. I am, as well, an amateur pianist.

1.1.1 Academic Background

In recognition of my academic achievement, I was sponsored by a Dean's Development Fellowship from New York University (NYU) and a teaching assistantship from New York Institute of Technology (NYtech). I started the joint program of NYtech/NYU in 1984, and finished my MS/Computer Science degree and post master applied science research in 1987. Since that time, I have undertaken informal academic training in the United States (1990-91). I made a visit to Imperial College in London (2003) for short-term research. I also joined a formal PhD by publication program at the Faculty of Information Technology at University Technology, Sydney (2006-07). Over the past two decades, I have been extensively involved in industry research by writing papers and leading industry innovation. I also have been highly engaged in thought leadership in enterprise architecture and integration. Examples of my innovations are event-driven technology, component-based project architecture (DCRP), common component identification, cursor computing, database synchronization, and metadata strategy. I have been involved in joint programs between academia and industry, such as giving lectures for universities and alliance/partners. It has also been my privilege to mentor numerous engineers and architects in professional development. Finally, I have published a high volume of technical articles in the IT community, and have applied research results to many IT projects.

1.1.2 Research Background

My research interests include enterprise architecture, integration, database, computing, and methodology. The significance of my research work is applying the approaches from my research directly to industry projects, generating huge business impact. I have had extensive exposure to consulting, advising, mentoring, and helping IT key vendors to maintain a high standard of thought leadership and operating governance. I enjoy sharing viewpoints with people as I trust knowledge is the origin of creativity, power, and order.

As I was fully sponsored by NYtech/NYU during my graduate study (1984-1987), I had much exposure to teaching and research. I frequently consulted with my advisor, Dr. B. Rugg, to apply partial differential equations in system modeling. The environment at NYU motivated my research interest as I very much enjoyed the academic atmosphere. From 1987 to the present I have taken advantage of many opportunities to conduct field research. In order to receive more formal education in research I joined the PhD by publication program of UTS in 2006. I perceive UTS to be a highly recognized university for its extensive industry research. From UTS, I have enjoyed a wider exposure to the academic world, which has helped my research and publications. I hope that my publications can bring significant contributions to both academia and industry. My first research paper, published in 1984, was entitled "Computation in Process Control and Automation." Since then, I have produced a high volume of technical writing and IT research publications in the field of enterprise integration. My publications have grown rapidly in recent years as I aim to contribute my experience to academia and industry. I believe publication is the best way of facilitating wider exposure for public access. I frequently find myself sitting up until 4 in the morning as I concentrate my mind on the process!

In recognition of my academic exposure, I have been requested by a few journals to act as a guest editor and reviewer in the field of IT architecture and service transformation. These activities help me to extend my exposure and experience in the role of coordinator between academia and industry. As a senior IT professional, I realize that this role is crucial and highly demanded. I have been glad to play a dual role in devoting myself to mediation activities between academia and industry, to improve the interaction between them.

1.1.3 Industry Background

The companies I have worked with have close linkage with academia. I have therefore had much opportunity to engage in research work by joining conferences, facilitating workshops, attending industry forums, and giving lectures. My current responsibility is to help IBM to enable enterprise architecture in service-on-demand in Greater China and APAC in the future. My position requires profound research in technology and enterprise, and requires significant innovation and knowledge sharing with my IBM colleagues worldwide. Basically the knowledge contribution from my publications helps global IT transformation and the IBM communities in developing cutting-edge technologies.

1.1.4 Motivation

My career vision is to play a role in mediating services between research and industry. To that end, I have been pursuing the goal of bridging research institute, university, and industry to generate more interplay. I am confident that I am taking a valuable step towards encapsulating my industry and academic background into a coherence of these two communities. I view the process as producing doubled interference values. The interactive model shown in Figure 1 is introduced to enhance the influence model. In the interactive model, reviews maintain a close interaction between academia and industry. This model is applied in this thesis by taking examples of issues and solutions from industry projects, to generate close linkage with research through reviews. Consequently, academia will root industry issues and industry will extend research to produce encapsulated effects.

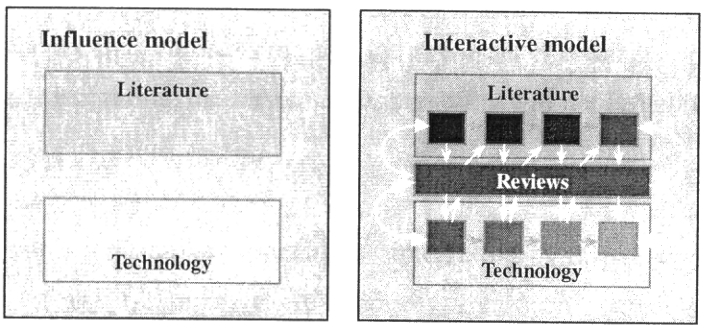


Figure 1. Comparing Interactive Model and Influence Model

1.2 Problem Identification and Key Contributions

1.2.1 Problem Identification

Before the introduction of new concepts, this thesis identifies the following problems as key issues in enterprise integration:

1) Misalignment of architecture, causing ineffective enterprise integration:
   This results in a weak responsiveness toward business opportunity. Many companies risk high impact or significant cost in focusing integration at system level instead of enterprise integration between virtual and physical architecture.

2) The failure of industry to effectively deploy a micro-view interoperability strategy:
   In consequence, a small issue can cause a huge impact in financial and operational concerns. The lack of micro interoperability causes difficulty in achieving reuse, and fewer implementation standards within enterprise integration. From an enterprise perspective, the weak governance of collaboration can push the company into silos in organization, process, and systems, and also into running a high business risk.

3) Lack of methodology and common practices to form the enterprise-scale skeleton:
   Failure to comply with industry standards results in management and cost issues, as each company has a different strategy for implementing a similar business scenario. Simultaneously, weakness in common practices causes duplicate effort and high integration risk in aspects such as integration of business, process, application, data, and systems between organizations.

4) Weakness in an interactive vision between micro and macro view:
   Micro architecture deals with component and its attributes, while macro architecture forms its substantial characteristics through the synthesis of elements' interference. If their interference is missing, the service interoperability will not be solid.

1.2.2 Contributions

This thesis contributes innovations in enterprise integration by taking a micro view of ontological research as its initiative. It produces significant business impact by transforming industry from patching solutions to agility, setting up a framework to encompass the synthesis process. It follows the nature of business and direction of architecture implementation to streamline the integration life cycle. Project risk, development effort, and integration complexity can be eliminated by taking the

approaches derived from this thesis into realization. The following paragraphs further explain how these innovations cause significant industry impact.

Coping with the high degree of complexity inherent in large-scale architecture will involve tremendous research into new concepts. It is as a result of this research that industry will be driven towards new-generation computing. The present thesis, therefore, undertakes to investigate the root issues and conduct ontological research in enterprise integration. Based on the ontology of raw elements, the concepts create service dimensions with linkage to business values. The service dimension represents the mediation between physical and virtual elements. It is a composite component in achieving a particular business scenario in use-case form.

At the conceptual level, this thesis contributes to enterprise integration by addressing the problems raised in the previous section:

1) Enterprise vertical alignment in macro view:

   In addressing problem 1, this thesis introduces a macro view of enterprise integration by using vertical alignment across business, service, and physical layers. Once this alignment is established, the impact between business and technology enables service mediation between the virtual and physical worlds. The present thesis, therefore, undertakes to investigate the root issues and conduct ontological research in enterprise integration. Based on the ontology of raw elements, the concepts create service dimensions with linkage to business values. The service dimension represents the mediation between physical and virtual elements. It is a composite component in achieving a particular business scenario in use-case form.

2) Component synthesis in micro view:

   In addressing problem 2, the micro view of this mediation addresses the synthesis of elements for service dimensions. This second concept aims to apply element synthesis in creating services and interoperability. The process requires a framework to encompass key activities. The objective of these activities is to create services and enterprise interoperability by taking a micro view of ontological research as its initiative. With a framework set up to encompass the synthesis process, significant business impact results as industry moves from patching solutions to agility. The process follows the nature of business and direction of architecture implementation

to streamline the integration life cycle. Project risk, development effort, and integration complexity can be eliminated by taking the approaches derived from this thesis into realization.

3) Enterprise skeleton and industry compliance:

   In addressing problem 3, the third concept moves the micro-view approach forward to form the enterprise skeleton and lift the micro-vision to macro-vision. It encapsulates the empirical method and patterns, together with service dimensions, to align with business context and produce coarse-grain services. As vertical interoperability is formed, the skeleton will become transparent for industry compliance purposes. The synthesis of coarse/fine-grain services creates opportunity in constructing the enterprise skeleton and brings in business benefits of lower risk and higher reuse. It also establishes a foundation for industry common practices and service virtualization.

4) An interactive vision between micro and macro view:

   The synthesis of coarse/fine-grain services creates opportunity in constructing the enterprise skeleton and brings in business benefits of lower risk and higher reuse. It also establishes a foundation for industry common practices and service virtualization. The concepts proposed by this thesis contribute originality of knowledge in enterprise integration. Examples of successful project experience are IBM SOA/FSS, Vodafone GITA-CRM/Billing, EDS CIS, Oracle OIA/TR, and Citibank CCS/APAC. The following scenarios are examples of successful deployment by taking one or more than one of the concepts into realization.

In the early 1990s, the banking system faced a major challenge in service transformation. The foundation of technology was not in a stable enough state to cope with business innovation. As a lead architect at Citibank, my proposal for regional technology was well taken by APAC CTO, Tsuei, and Chairman, T. Mckeon. The proposal targeted parallelism of eliminating random access across APAC, as well as metadata-driven technology. It indicated that an enterprise alignment should be set as priority across the region. The proposal further recommended applying the concept of single source of truth for the goal of service virtualization. Strategies directed at maintaining a high semantics integrity and fragmentation-free environment were established for distributed computing. These strategies helped many projects be launched much more smoothly and quickly.

A second example is the telecom scenario in the early 2000s. As an enterprise architect of Vodafone in Japan, my major responsibility was to support global IT architecture compliance (GITA) in Japan. The goal was to bring key operators (OpCos) in line with the Vodafone GITA and enhanced Telecom Operations Map (eTOM). However, as many foundation prerequisites were missing due to historical merging and acquiring, it was difficult to move business strategy forward to the realization stage. These missing components included single view of customer, synchronization between CRM (customer relationship management) and billing, and global technology compliance. The proposal targeted component refining and applied pattern benchmarking in producing integration strategy. It convinced Vodafone Japan stakeholders that in order to comply with GITA, enterprise vertical integration was required to produce virtual service components. Meanwhile, service approaches were targeted to escalate horizontal compliance across global architecture. The recommendation was highly recognized by general manager P. Thrush and the architecture team. It made global compliance feasible by using well-defined services with well-specified semantics sharable across global branches for quick development.

Recently, my IBM colleagues in Germany had extensively discussed my publications with me. The concepts have been employed to streamline IBM cutting-edge technologies such as SOA, high availability & disaster recovery (HADR), and service on demand.

1.3 Inter-chapter Interference and Coherence

1.3.1 Paper Structure Aligned to Research Coherence

The structure of this thesis is arranged in the following order:

Chapter 0: Abstract
Chapter 1: Introduction
Chapter 2: Literature review
Chapter 3: Concept
Chapter 4: Strategy
Chapter 5: Framework
Chapter 6: Summary
Appendix: Publications

The research coherence and paper structure of this thesis is shown in Figure 2. The contents of the thesis may be seen to address two main areas, initiatives and realization. Initiatives take steps of the background, motivation, literature review, and concept, to derive strategy. Realization focuses on the META framework. The five blue labels denote thesis chapters, while the six red labels denote the categories of publications.

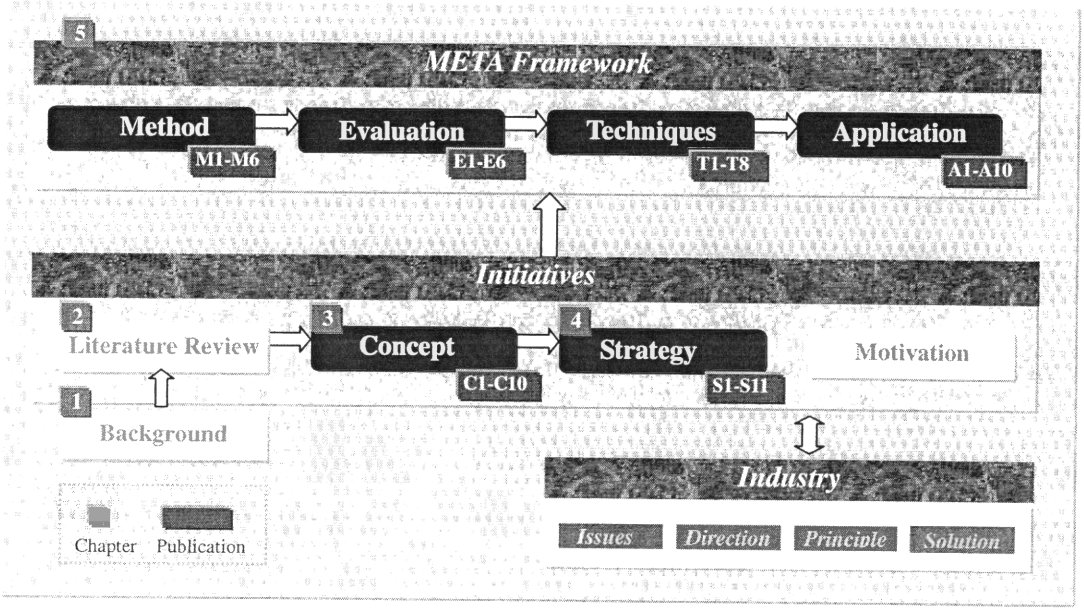


Figure 2. Coherence of Thesis Chapters and Publications

1.3.2 Statistics of Publications

The publications consist of 51 papers grouped in six categories, denoted by Concept (C), Strategy (S), Method (M), Evaluation (E), Techniques (T), and Application (A). Each paper is assigned to a primary category to extend the research in that category, though the paper may also relate to one or more of the other categories as well. The statistics of the 51 papers attached in the Appendices are summarized as follows:

Journal	Conference & Industry	Sum
12	39	51

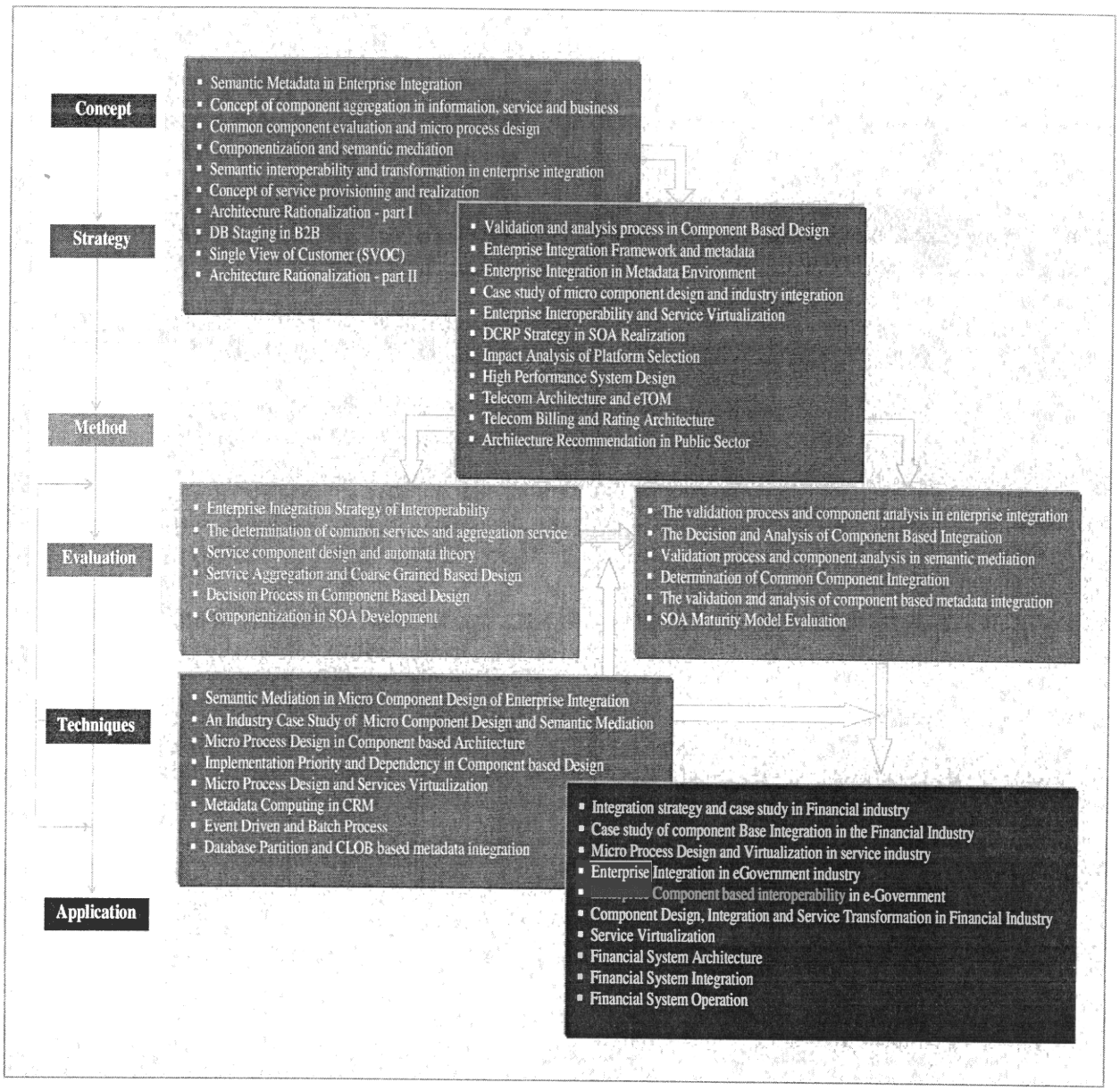
Table 1. Publication Statistics

1.3.3 Research Flowchart

Enterprise integration is such a huge topic that the coherence of the research cannot properly rely on an integrated approach which develops subsets into a whole. Instead, this thesis takes the approach of starting with an abstract of research overview and issue identification to derive the strategy. The strategy takes both micro and macro views to root the common defects and proposes approaches in constructing the framework. The enterprise approach using the META framework is to develop a validation process to identify and specify atomic elements. It also develops synthesis processes to design the micro interconnectivities for service patterns and semantics. The interference which derives strategy into this framework produces the cornerstones of enterprise integration. From a publication perspective, this thesis consists of the following six sections.

- Concept: Enterprise Vertical Integration, Three Steps and Four Pillars, and Synthesis
- Strategy: Metadata, Semantics, Interoperability, Common Components, and Framework
- Method: Componentization, Ontology, Validation, Political and Analysis Model
- Evaluation: Component Analysis and Evaluation, Parsing, Aggregation, and Synthesis
- Techniques: Semantic Mediation, Micro/Macro Process, Parsing, and Aggregation
- Application: Virtualization, Industry Realization, Transparent, and Common Services

The following flowchart shows the interference between sections; each section consists of a list of publication, and the sum of the lists produces a total of 51 papers.


Concept
Strategy
Method
Evaluation
Techniques
Application
Semantic Metadata in Enterprise Integration
Concept of component aggregation in information, service and business
Common component evaluation and micro process design
Componentization and semantic mediation
Semantic interoperability and transformation in enterprise integration
Concept of service provisioning and realization
Architecture Rationalization - part I
DB Staging in B2B
Single View of Customer (SVOC)
Architecture Rationalization - part II
Validation and analysis process in Component Based Design
Enterprise Integration Framework and metadata
Enterprise Integration in Metadata Environment
Case study of micro component design and industry integration
Enterprise Interoperability and Service Virtualization
DCRP Strategy in SOA Realization
Impact Analysis of Platform Selection
High Performance System Design
Telecom Architecture and eTOM
Telecom Billing and Rating Architecture
Architecture Recommendation in Public Sector
Enterprise Integration Strategy of Interoperability
The determination of common services and aggregation service
Service component design and automata theory
Service Aggregation and Coarse Grained Based Design
Decision Process in Component Based Design
Componentization in SOA Development
The validation process and component analysis in enterprise integration
The Decision and Analysis of Component Based Integration
Validation process and component analysis in semantic mediation
Determination of Common Component Integration
The validation and analysis of component based metadata integration
SOA Maturity Model Evaluation
Semantic Mediation in Micro Component Design of Enterprise Integration
An Industry Case Study of Micro Component Design and Semantic Mediation
Micro Process Design in Component based Architecture
Implementation Priority and Dependency in Component based Design
Micro Process Design and Services Virtualization
Metadata Computing in CRM
Event Driven and Batch Process
Database Partition and CLOB based metadata integration
Integration strategy and case study in Financial industry
Case study of component Base Integration in the Financial Industry
Micro Process Design and Virtualization in service industry
Enterprise Integration in eGovernment industry
Component based interoperability in e-Government
Component Design, Integration and Service Transformation in Financial Industry
Service Virtualization
Financial System Architecture
Financial System Integration
Financial System Operation

1.4 List of Publications

1.4.1 Publications in academic format

✧ Journals Publication

Wu, R., and Lu, J. 2008, 'The Validation Process and Component Analysis in Enterprise Integration', *International Journal of Business and Systems Research (IJBSR)*, vol.2, issue 1

Wu, R. 2008, 'Semantic Mediation Modeling in the Service Component Design', *International Journal of Metadata, Semantics and Ontology (IJMSO)*, vol.3, issue 1

Wu, R. 2008, 'An Industry Case Study of Micro Component Design and Semantic Mediation', *International Journal of Metadata, Semantics and Ontology (IJMSO)*, vol.3, issue 2

Wu, R. 2008, 'Enterprise Interoperability and Service Virtualization', *Association for Computing Machinery New Zealand Bulletin (ACM_NZ)*, vol. 4, issue 1 [online], available from w.w.w: *http://www.acm.org/chapters/acm_nz/bulletin*

Wu, R. 2008, 'Case Study of Micro Component Design and Industry Integration', *International Journal of Intercultural Information Management (IJIIM), vol.1, issue 3*

Wu, R. 2007, 'Enterprise Integration Framework and Metadata', *International Journal of Intelligent Information and Database Systems (IJIIDS)*, vol. 1, issue 3

Wu, R. 2007, 'Component Design, Integration and Service Transformation in Financial Industry', *International Journal of Information Systems and Change Management (IJISCM)*, vol. 2, issue 2, pp.155-166

Wu, R. 2007, 'Enterprise Integration in Metadata Environment', Journal of Object Technology *(JOT)*, vol. 6, no. 10, paper no. 6

Wu, R. 2007, 'Enterprise Integration in e-Government', *Electronic Government, an International Journal (EG)*, vol. 4, issue 2, pp.179-190

Wu, R. 2007, 'Determination of Common Component Integration', *Association for Computing Machinery New Zealand Bulletin (ACM_NZ)*, vol. 3, issue 1 [online], available from w.w.w: *http://www.acm.org/chapters/acm_nz/bulletin, paper no. 3*

Wu, R. 2007, 'Decision Process in Component Based Design', *Association for Computing Machinery New Zealand Bulletin (ACM_NZ)*, vol. 3, issue 2 [online], available from w.w.w: *http://www.acm.org/chapters/acm_nz/bulletin, paper no. 3*

Wu, R. 2007, 'Semantic Interoperability and Transformation in Enterprise Integration', *Association for Computing Machinery New Zealand Bulletin (ACM_NZ)*, vol. 3, issue 2 [online], available from w.w.w: *http://www.acm.org/chapters/acm_nz/bulletin, paper no. 2*

✧ Conferences and Industry Publication

Wu, R. 2007, 'Implementation Priority and Dependency in Component Based Design', July 2007, IEEE CS/EISWT, Orlando, USA, 9-12 July 2007, pp 42-46

Wu, R. 2007, 'Common Component Evaluation and Micro Process Design', July 2007, IEEE CS/EISWT, Orlando, USA, 9-12 July 2007, pp 47-52

Wu, R. 2007, 'The Determination of Common Services and Aggregation Service', July 2007, IEEE CS/EISWT, Orlando, USA, 9-12 July 2007, pp 36-41

Wu, R. 2007, 'Service Component Design and Automata theory', July 2007, IEEE CS/EISWT, Orlando, USA, 9-12 July 2007, pp 53-57

Wu, R. 2007, 'Validation process and Component Analysis in Semantic Mediation', IEEE/ICIS, Melbourne, Australia, 11-13 July, 2007, pp 873-878

Wu, R. 2007, 'The Decision and Analysis of Component Based Integration', The 11th International Conference on Human Aspects of Advanced Manufacturing Agility and Hybrid Automation (Haamaha), Poznan, Poland , 9-12 July, 2007, Modern Enterprise paper no. 6

Wu, R. 2007, 'Micro Process Design in Component based Architecture', , The 11th International Conference on Human Aspects of Advanced Manufacturing Agility and Hybrid Automation (Haamaha), Poznan, Poland , 9-12 July, 2007, Modern Enterprise paper no. 4

Wu, R. 2007, 'Enterprise Component based interoperability in e-Government', The 11th International Conference on Human Aspects of Advanced Manufacturing Agility and Hybrid Automation (Haamaha), Poznan, Poland , 9-12 July, 2007, Modern Enterprise paper no. 10

Wu, R. 2007, 'Concept of component aggregation in information service and business', IEEE/ICSSSM, Chengdu China, 9-11 June, 2007, pp. 626-629

Wu, R. 2007, 'Service Aggregation and Coarse Grained Based Design', IEEE/ICSSSM, Chengdu China, 9-11 June, 2007, pp. 682 – 685

Wu, R. 2007, 'The validation and analysis of component based metadata integration', ACM/EATIS ACM-DL Proceedings, Faro, Portugal, 14-17 May 2007, paper no.29

Wu, R. 2007, 'Micro Process Design and Virtualization in service industry', ACM/EATIS, ACM-DL Proceedings, Faro, Portugal 14-17 May 2007, paper no.29

Wu, R. 2007, 'Validation and analysis process in Component Based Design', IEEE/SoutheastCon, Richmond VA, USA, 22-25 March, 2007, pp 208-212

Wu, R. 2007, 'Micro Process Design and Services Virtualization', IEEE/SoutheastCon, Richmond VA, USA, 22-25 March, 2007, pp 213-216

Wu, R. 2007, 'Integration strategy and case study in Financial industry', IEEE/AMS, Thailand, March 2007, pp 384-387

Wu, R. 2007, 'Case study of component Base Integration in the Industry', IEEE/SoutheastCon, Richmond VA, USA, 22-25 March, 2007, pp 243-246

Wu, R. 2007, 'Semantic Metadata in Enterprise Integration', IEEE/SoutheastCon, Richmond VA, USA, 22-25 March, 2007, pp 233-236

Wu, R. 2007, 'Componentization and semantic mediation', November 2007, IECON2007, paper no. TD-018309

Wu, R. 2007, 'Concept of Service provisioning and_realization', International Conference on Intelligent Systems and Knowledge Engineering (ISKE2007), Chengdu, China, Oct 15-16, 2007, pp 245-248

Wu, R. & Lu, J. 2006, 'Enterprise Integration Strategy of Interoperability', IEEE/CISSE online conference, 4-14 December, 2006, SCSS paper no. 56

Wu, R. 2006, 'DCRP strategy in SOA realization', IBM Software SWITA Compliance

Wu, R. 2006, 'Infrastructure Rationalization', EDS APAC Technology Conference

Wu, R. 2006, 'Componentization in SOA development, ' EDS CTC Presentation

Wu, R. 2006, 'Single View of Customer (SVOC)', IBM Enterprise Architecture presentation

Wu, R. 2006, 'SOA maturity model evaluation', IBM Enterprise Architecture Presentation

Wu, R. 2006, 'Service Virtualization', IBM Enterprise Architecture Presentation

Wu, R. 2006, 'Architecture Rationalization - part 1', EDS APAC Technology Conference

Wu, R. 2006, 'High Performance System Design', IBM Software Group Architecture Compliance

Wu, R. 2006, 'Architecture recommendation in Public Sector', , IBM Software Group Presentation

Wu, R. 2005, 'Financial System Architecture', Oracle Emerging Technology

Wu, R. 2005, 'Financial System Integration', Oracle Emerging Technology

Wu, R. 2005, 'Financial System Operation', Oracle Emerging Technology

Wu, R. 2004, 'Metadata Computing in CRM', Oracle Alliance Work Shop

Wu, R. 2004, 'Event driven and Batch Process', Oracle Alliance Work Shop

Wu, R. 2003, 'Impact Analysis of Platform Selection', Vodafone GITA Presentation

Wu, R. 2003, 'Database Partition and CLOB', Oracle Alliance Work Shop

Wu, R. 2003, 'Telecom architecture and eTOM', EBT/Oracle Alliance Presentation

Wu, R. 2003, 'Telecom Billing & Rating Architecture', Vodafone GITA Presentation

Wu, R. 2002, 'DB Staging in B2B', Portal Japan Board Committee

1.4.2 Publications Sorted by Category

Papers List (Table 2)

Journal | Conference | Industry

ID	Title	Publisher/Conf	Page
	Concept		
C1	*Semantic Metadata in Enterprise Integration*	IEEE-Southeast	74
C2	*Concept of component aggregation in information, service and business*	IEEE-ICSSSM	80
C3	*Common component evaluation and micro process design*	IEEE-EISWT	87
C4	*Componentization and semantic mediation*	IEEE-IECON	97
C5	*Semantic interoperability and transformation in enterprise integration*	ACM_NZ	105
C6	*Concept of service provisioning and realization*	ISKE-2007	113
C7	*Architecture Rationalization - part I*	EDS Corp.	120
C8	*DB Staging in B2B*	Portal Japan	127
C9	*Single View of Customer (SVOC)*	IBM Corp.	134
C10	*Architecture Rationalization - part II*	EDS Corp.	143
	Strategy		
S1	*Validation and analysis process in Component Based Design*	IEEE-Southeast	150
S2	*Enterprise Integration Framework and metadata*	IJIIDS	157
S3	*Enterprise Integration in Metadata Environment*	JOT	168
S4	*Case study of micro component design and industry integration*	IJIIM	175
S5	*Enterprise Interoperability and Service Virtualization*	ACM_NZ	185
S6	*DCRP Strategy in SOA Realization*	IBM Corp.	198
S7	*Impact Analysis of Platform Selection*	Vodafone Japan	206
S8	*High Performance System Design*	IBM Corp.	215
S9	*Telecom Architecture and eTOM*	Oracle Corp.	222
S10	*Telecom Billing and Rating Architecture*	Vodafone Japan	229
S11	*Architecture Recommendation in Public Sector*	IBM Corp.	235
	Method		
M1	*Enterprise Integration Strategy of Interoperability*	IEEE-CISSE	241
M2	*The determination of common services and aggregation service*	IEEE-EISWT	251
M3	*Service component design and automata theory*	IEEE-EISWT	260
M4	*Service Aggregation and Coarse Grained Based Design*	IEEE-ICSSSM	268
M5	*Decision Process in Component Based Design*	ACM_NZ	275
M6	*Componentization in SOA Development*	EDS Corp.	285
	Evaluation		
E1	*The validation process and component analysis in enterprise integration*	IJBSR	293
E2	*The Decision and Analysis of Component Based Integration*	Haamaha	312
E3	*Validation process and component analysis in semantic mediation*	IEEE-ICIS	326
E4	*Determination of Common Component Integration*	ACM_NZ	334

E5	*The validation and analysis of component based metadata integration*	ACM-EATIS	344
E6	*SOA Maturity Model Evaluation*	IBM Corp.	352
	Techniques		
T1	*Semantic Mediation in Micro Component Design of Enterprise Integration*	IJMSO	357
T2	*An Industry Case Study of Micro Component Design and Semantic Mediation*	IJMSO	376
T3	*Micro Process Design in Component based Architecture*	Haamaha	394
T4	*Implementation Priority and Dependency in Component based Design*	IEEE-EISWT	407
T5	*Micro Process Design and Services Virtualization*	IEEE-Southeast	416
T6	*Metadata Computing in CRM*	Oracle Corp.	421
T7	*Event Driven and Batch Process*	Oracle Corp.	427
T8	*Database Partition and CLOB*	Oracle Corp.	432
	Application		
A1	*Integration strategy and case study in Financial industry*	IEEE-AMS	440
A2	*Case study of component Base Integration in the Financial Industry*	IEEE-Southeast	446
A3	*Micro Process Design and Virtualization in service industry*	ACM-EATIS	452
A4	*Enterprise Integration in eGovernment industry*	EG Inderscience	462
A5	*Enterprise Component based interoperability in e-Government*	Haamaha	476
A6	*Component Design, Integration and Service Transformation in Financial Industry*	IJISCM	486
A7	*Service Virtualization*	IBM Corp.	499
A8	*Financial System Architecture*	Oracle Corp.	504
A9	*Financial System Integration*	Oracle Corp.	513
A10	*Financial System Operation*	Oracle Corp.	522

CHAPTER 2

LITERATURE REVIEW

2.1 Enterprise Integration Issues

The IT industry became pervasive in this decade due to its significant growth in terms of high-volume transactions and sophisticated businesses. According to IBM (2006), 78% of Chief Executive Officers (CEOs) believe integrating business and technology is fundamental for survival and innovation. Moreover, only less than 45% of CEOs believe their organization has the ability to be responsive to changing business conditions. These figures imply that over 42% of CEOs would rank enterprise integration as a top priority since they are not ready to face this challenge. In order to meet this need, this thesis focuses on fundamental research in addressing this issue.

2.1.1 Architecture Misalignment

This thesis proposes vertical alignment between business, service, and physical components. The issues the approach addresses have been prominent for the last decade and their business impact has been ranked top among enterprise-level issues (Wu, 2006). DSouza (2001) proposed vertical integration, stating that it comes from different levels of abstraction of the same system. DSouza also defined data models, networks, applications specifications, component assemblies, business process models, and business strategies with their interferences in the metadata layer. IBM (2005) stated that each component has its contract-level responsibility in provisioning a subset of services. The resulting interface helps the coordination between components and metadata build the foundation of interoperability within the service layer and between layers.

Both of these theories are relevant to the approaches this thesis proposes, though the literature is lacking in semantics strategy. Furthermore, the micro process and realization are not clearly outlined. In a vertical hierarchy, the upper layer is the abstract of aggregation from lower layers instead of a one-to-one object mapping. The higher layer plays a more important role in mediating human aspects for those contexts and semantics (Petrie, 2007). This concept replaces conventional approaches as it enables mediation between virtual and physical instead of the traditional "glued" type of integration between layers or systems.

2.1.2 Issues in Micro View and Semantic Interoperability

Hadzilakos et al. (2000) clarified enterprise interoperability as the key concept for integrating various kinds of information stored in a variety of repositories. However, interoperability is absent in enterprise integration due to incompatible software, inadequate data formats, semantic heterogeneity (Zhu, 2005), data model, and metadata (Pinto et al., 2003). It is believed that these factors are the major causes for a semantics gap (Bright, Hurson, & Pakzad, 1994; Uschold & Gruninger, 2004).

Service transformation aiming at interoperability from a technical perspective is common in the industry. Based on the analysis of Selvage et al. (2006), there are several different implementation patterns to achieve semantic interoperability. These include point-to-point semantic integration (Mackay, 1999); hub-and-spoke semantic integration; master data management (MDM) (Bergamaschi, Castano, & Vincini, 1999); the industry information model; and the semantic web (Goble & Roure, 2002). However, from the viewpoint of human factors such as semantics, context, and metadata, a pure technique is not sufficient. For this reason, enterprise interoperability is not always achievable in taking business semantics into technical implementation (Nord, 2007; Petrie, 2006).

Semantics and interoperability, according to Kruchten (1998), are all the means to attempt to homogenize disparities through standards. Following Kruchten, some other literature applied semantics in interfacing dispersed systems into homogeneity (Al-Masri & Mahmoud, 2007). From the organization perspective, a number of researchers declared that semantics strategy is how people identify and possibly resolve their semantic differences (Kajan & Stoimenov, 2005). A reason for this is that a geographically heterogeneous state in semantics forces peers to follow a power law distribution.

Vckovski, Brassel, & Schek (1999) are recognized as almost the first researchers to define interoperability. Their definition is the ability to exchange and integrate meaningful information. Harvey et al. (1999) defined semantic interoperability from a human factors perspective. Their definition involves looking at how people and social groups with different perspectives identify and possibly resolve their semantic differences but lack in resolution. Harvey also proposed the need for frameworks (Kajan, 2004). Frameworks are called for in the construction of information communities in

heterogeneous institutional and disciplinary environments. The main purpose of such frameworks is to conceptualize and articulate these semantic differences (Ouksel & Sheth, 1999; Smith & Obrst, 1999). In addition, meta-level object definition and single source of truth are priorities in building up an efficient architecture (Maedche, Motik, & Stojanovic, 2003).

To improve interoperability efficiency, the integration trend in IT has changed recently from peer-to-peer (n:n) to hub and spoke (1:n), and from spoke to service bus (1:1) (Tolosana-Calasanz et al., 2006). Batini, Grosso, & Longobardi (2006) pointed out that the pre-requirements for this restructuring are the consolidation of all duplicate objects and data fragmentation toward single source of truth. This can be resolved by means of aggregation or merging toward achieving a meta-level of master data. Thus interoperability can be attained within the meta-layer to effectively impact the individual entity (Aslan & McLeod, 1999; Schiavone, Sureshchandran, & Hardis, 1997). Sicilia (2006) pointed out that the fact that metadata is created to support some specific function is sometimes overlooked or vaguely acknowledged. The issue is that in some situations metadata does not transmit semantics effectively to support loosely-coupled approach, and requires special handling. Therefore, interference between semantics and metadata is crucial in interoperability however, exceptions need be managed (Aslan & McLeod).

2.1.3 Weakness in Common Practices and Industry Framework

This thesis indicates the absence of vertical alignment as the root issue of industry chaos. It predicts that enterprise vertical integration will become mainstream in industry in next-generation computing. Supporting this prediction is the intention already demonstrated by several industry key vendors of strategic compliance with the concepts proposed in this thesis. By taking the concept of vertical integration into strategy, this research applies META approach to strengthen the foundation in realization. Figure 3 illustrates the sequence and interference of the four META pillars. It indicates that the interference within META does not necessarily follow a particular sequence. For instance, the ontology produced in the validation stage may impact applications directly if a lower service maturity level is targeted. The proposed META framework aims to encompass enterprise integration to cope with issues such as chaotic business change, dispersed semantics, and system silos. Lytras (2006) introduced an enterprise framework to build a common foundation, a concept which doesn't, however, work effectively on an

enterprise scale. Therefore this thesis will only refer to that work in broad terms instead of going into the details of the approach proposed by Lytras.

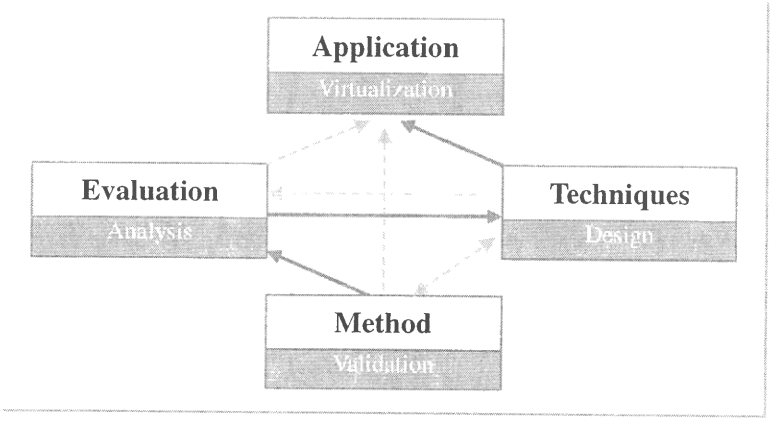


Figure 3. Four Pillars of META Framework

Among the four pillars of the META framework this thesis proposes, the first step of the META approach is the method for component validation. The scope of the validation process is ensuring that all the required elements exist in the ontology of elements. It doesn't cover service design; however, it does concern technical specification of elements, element attributes, and ontological grouping. The objective of validation is to identify the candidacy of atomic elements for refining synthesis, in producing common service and service aggregation. This is somewhat similar to the approach proposed by Scha, Bod, and Sima'an (1999). In componentization, firstly, the micro process deconstruction is initiated by a domain analysis in order to form the enterprise elements as domain assets (Hunter, 2001). This collection of enterprise elements in atomic form is termed the ontology of elements in this thesis (Bunge, 1977; Bunge, 1979). In addition, this approach applies incremental and parallel design in componentization. The idea here is that compatible elements and interfaces can be refined separately while still maintaining compatibility (Alfaro & Henzinger, 2001). From a decomposition viewpoint, this thesis proposes a top-down approach as a natural method of business deconstruction (Gosain, 2003). From a mediation viewpoint, the bottom-up approach is proposed to take an abstract of physical components into service component design in developing semantic interoperability. Service components derived from the business components, as well as from physical components are the mediation between business and technical

elements. Further refining synthesis within service components is required as mediation aims to integrate all business semantics, interfaces, and technical configurations into a virtual design (Harth, 2004). In order to construct enterprise interoperability from both micro and macro view, initially all the components are at an atomic level (Maedche et al., 2001; Valkeapää & Hyvönen, 2006). In this way we start with an ontology of elements in atomic form.

The process following component validation is service analysis and evaluation. The decision-making process uses group decision-making to take each individual opinion or preference and multiply it by the role weight factor (Lu, Zhang, & Wu, 2005). This thesis uses a similar political model approach but uses different weights for different roles and different evaluation scopes. The objective of evaluation is to create priority and visibility for service component design. This thesis proposes service synthesis to transform the raw elements into common components and aggregation of services. Through the use of parsing techniques, service dimensions are derived and atomic elements are encapsulated to virtual dimensions in fulfilling mediation requirements. The coarse-grained service developed in the synthesis process provides upward mediation to the business layer. It produces advantages in eliminating the integration interface caused by boundary implementation (Dwarkadas et al., 1999). As coarse-grained service is critical in the enterprise skeleton, it should be strictly moved through the processes of identification, validation, analysis and evaluation (US Industry Advisory Council, 2003).

Both "aggregation" and "common" should not take a tightly coupled approach in interface design. Instead, their dependency and interconnectivity should be supported by semantics and be implemented on loosely coupled metadata approaches. Also important is that the component design and interfacing specifications be fully visible and stored in repositories for future development (Curtis, Foster, & Stentiford, 1999). Vernadat (1997) pointed out that enterprise modeling is a required process in those common components that can be reused. O'Brien et al. (2004) further pointed out that interoperability supports the collaboration of business partners as they produce value and services in supply chain management. This implies that once the vertical service components are implemented, the provisioning should be extended across the industry for horizontal compliance.

2.2 Unique Characteristics in Semantic Enterprise Integration

The key achievements by which this thesis distinguishes itself from existing literature can be identified as follows:

Currently there are very few studies that address real enterprise-scale integration, and none from our finding cope with fundamental research in root issues. Most of the literature covers only add-on enhancements and is unable to settle real issues from the viewpoint of both scale and complexity. By contrast, the research presented in this thesis develops semantic interoperability in micro view integration and sets up a framework for realization (Mirhaji et al., 2007). It aims to achieve service virtualization through a vertical alignment using a multi-discipline approach. It also develops a META approach by projecting Service Component Interoperability Strategy (SCIS) to the life cycle of validation and analysis (Benjamin et al., 2005; Gray, Findler, & Flatt, 2005; Lam & Shankararaman, 2004). This process necessitates the construction and application of a number of specific techniques and ontological approaches.

Most of the literature in interoperability has taken a "patching" approach. This approach is based on the assumption that the semantics gap can be filled in by using a common data model, system interfacing, or through application integration. This assumption may be misleading for enterprise. There is a common misunderstanding that by developing tools and a common database, the plug-in packages can effectively resolve enterprise integration. This thesis distinguishes itself in its interoperability insight and indicates that the real issue is "velocity" instead of merely "current state". In other words, one's crucial responsibility, as an enterprise integration researcher, is to lead the industry in strengthening the industry mainstream with a common foundation. In this way the pre-built portfolio can dynamically tune its direction in coping with a chaotic world. This is particularly important given the exponential growth of business innovation and service transformation. For this reason, a solid foundation together with capability and methodology for coping with agility are proposed as the focus of enterprise integration. This thesis focuses on business values and semantics as initiatives to produce macro linkage between business and technology. It also concerns micro synthesis in taking semantics approaches into enterprise integration coherence.

Semantics interoperability approaches proposed in this research distinguishes itself from

conventional theory for the following reasons. This thesis proposes a three-level derivation: a) technical level in element format; b) syntax level in component interconnectivity (Botting, 2005); and c) context level leveraged to the business layer. So far this three-level semantics coherence has not been addressed in the literature review. This is cause for concern since none of these three levels can be omitted if one is to take human factors and enterprise vision into interoperability coherence. The approach of this thesis is to envision micro and macro synthesis in semantics. So each service component will have connection with business, service, and technical layers, through semantics and metadata strategy, in achieving a vertical interoperability.

2.3 Originality of Knowledge Contribution

2.3.1 Cornerstones of Enterprise Integration

Many researchers have identified enterprise integration as a top issue because the scale and complexity of new-generation computing is beyond the capacity of this decade. However, most of the literature is still stuck on patching solutions which cannot be integrated into an enterprise vision. Consequently, many patching solutions don't form integrated approaches as a whole to cope with complex issues. Many individual disciplines fail to achieve interoperability coherence as well. When enterprise acts without common practices, and implementation uses varied approaches, the result is semantic conflict. This conflict causes dispersed service interfaces and system silos. Sequence, priority, and enterprise collaboration may fail when enterprise cannot project its strategy into the industry mainstream (Joannou, 2007).

This thesis takes business initiative by proposing a top-down approach and introduces concept, methods, strategy, and framework to envision the multi-discipline interference. The cornerstone of the approach is the use of a simple concept to encompass strategy within a framework. The concept leverages to business values, as the goal of enterprise interoperability is to achieve a business target instead of merely making systems "integrated." However, to achieve business goals in an increasingly complex environment, business and technology require a mechanism to synchronize their parallelism of acceleration. The use of mediators becomes crucial, and the focus must be on a robust vector instead of static patching. This vector is produced by an "engine" which is actually the framework itself. The engine sources element ontology and produces a variety of mediators according to business direction and change. The strategy

breaks down how to specify and produce mediators. So when mediators extend their interoperability coherence into the service layer, the value of mediation is not merely limited to a business semantic as "object." Instead it achieves the business goal as a "vector." In other words, it dynamically acts on semantic responsiveness and mediation in a chaotic world of business innovations and emerging technologies (Gleghorn, 2005).

2.3.2 Concept of Enterprise Vertical Integration

Just as in component-based design, analysis, validation, vertical alignment, and service synthesis are all key processes in producing interoperability. This thesis further explains how the concept of vertical alignment relates to the enterprise skeleton. The enterprise skeleton is actually the aggregation of common components, the enterprise common services which need be identified and synthesized. The approach described in this thesis has been implemented in several large accounts. It has helped a major insurance service provider to build an enterprise skeleton using hundreds of common components produced by the indicated framework. In addition to common services, the metadata repository also contributes vertical alignment. It forms the enterprise skeleton and enables the enterprise to extend the skeleton into industry compliance (Papazoglou & Heuvel, 2007; Sicilia et al., 2005).

2.3.3 Strategy and Foundation for Enterprise Interoperability

Based on the strategy and framework developed in this thesis, the originality of knowledge contribution can be categorized into five aspects. These are ontological analysis, the synthesis of micro/macro interference, the synthesis of semantic interoperability, the vector concept in service pattern design, and a multi-discipline approach.

1) Ontological analysis: An anatomy of the ontology of enterprise integration is conducted in producing a three-dimensional model: space, time, and service. The space dimension denotes a scalable element or technical asset. The time dimension refers to the step of the life cycle or sequence in component design. The value dimension has to do with a business context or with service patterns in the virtual world. The interference between these three dimensions forms a focus in semantic coherence of service transformation.

2) Synthesis of micro/macro interference: The interoperability coherence is not addressed in the existing literature. For this reason, this thesis proposes a

micro/macro synthesis, that is, a synthesis between semantics and metadata. It positions the individual step in the full life cycle when the three-dimensional model is taken into realization. It addresses alignment as well, for how a business value aligns with interoperability.

3) Synthesis of semantic interoperability: Component validation and analysis constitute a synthesis process, which is one of the keys of the framework. This thesis further benchmarks semantic strategy and services patterns in supporting synthesis process, since different patterns require different synthesis approaches. In order to encompass high-complexity architecture in an environment of rapid development, an enriched-service pattern is proposed. This pattern applies a fine/coarse-grain approach and robust semantic strategy to construct the enterprise skeleton.

4) Service is a vector rather than an object. This study proposes the "vector" concept to build up agility in coping with a chaotic world. More precisely, vector means the responsiveness of capability toward business goal. It reaching the goal through service dimension design, instead of patching on the existing environment.

5) Multi-discipline in constructing the framework: The concern, therefore, is to construct a multi-discipline approach. Among these disciplines, semantic mediation is a key in holding interoperability coherence (Rosenthal, Seligman, & Renner, 2004).

2.3.4 Synthesis of Fine/Coarse-Grain Enriched Services

As has been indicated in section 1.2.2, the synthesis of fine/coarse-grain components and semantics strategy produce a service portfolio in mediating business and technology. Service components are not evenly distributed in the service layer in terms of grain size and semantics. Therefore the abstract of semantics and grain size aggregates in an upward direction. This thesis develops a map produced by plotting the Common Components and Service Aggregation (CCSA). Each service component, when plotted against the CCSA map, may qualify for service aggregation if it follows a certain sequence with a certain frequency. That is, two or more atomic service components can be aggregated upward to a coarse-grained service. The grain size and threshold are determined by the architecture group. The architects in the decision group use either empirical benchmarking or optimization desciplines in their evaluation. They will

examine advantage (i.e., coarse grain helps in minimizing interfaces) and disadvantage (i.e., loss of flexibility in the new configuration).

2.3.5 Scenarios of Business Impact

The following scenarios are examples of successful deployment by taking one or more of the indicated concepts into realization. The concepts have contributed originality of knowledge in enterprise integration and caused variety of business impacts based on the integration scenarios. .

Contribution vs. scenario-values	Architecture alignment	Micro view interoperability	Common service and component	Industry compliance
Hosting services (IBM)			Common, lower cost	
e-Government NP (IBM)		Reuse & lower risk	Common, lower cost	
Global arch. (Vodafone)	Vertical alignment		Common, lower risk	Standard
Insurance core (EDS)			Lower cost	Lower effort
Supply chain (Oracle)			Common, lower cost	Lower effort
Maritime (US government)			Global-local arch.	Standard & secured

Table 3. Examples of Business Scenarios and Contribution

CHAPTER 3

CONCEPTS

3.1 The Nature of Integration Ontology

Enterprise integration has been ranked as one of the top issues and one of highest complexity, as the aspect it targets is neither purely virtual nor purely physical. Instead it deals with the intermediate stage of the collaboration between virtual and physical. For this reason, an ontological investigation is conducted to shed light on the staging area. Figure 4 shows the origin of integration ontology for how this intermediate stage produces atomic elements and attributes. In the ontological representation symbolic criteria are used. These criteria are element and asset as elements; and grain size, roadmap, life cycle, velocity, event, sequence, motivation, semantics, and optimization as attributes. The ontology further groups the criteria into the three dimensions of space, time, and value by the nature of characteristics. This three-dimensional grouping, which exactly envisions the nature of integration ontology, represents the intermediate stage in the following format:

- Space: scalable element, technical asset, geography, and component grain size.
- Time: roadmap, step of the life cycle, velocity, event, and sequence
- Value: business context, motivation, semantics, dimension, vector, and optimization.

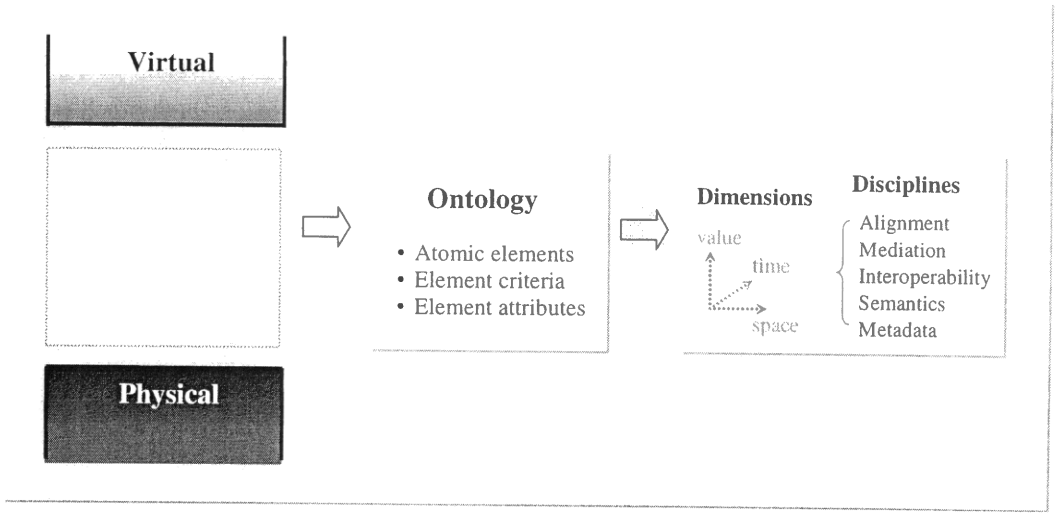


Figure 4. Origin of Integration Ontology

Thus, the staging layer of interest to this thesis can be projected on a matrix of ontology involving the three-dimensional interference of space, time, and business values. The ontological analysis of this thesis is based on an anatomy of the nature of integration itself, and elements constitute the interoperability. The research shows that instead of splitting the ontology into virtual and physical assets, the atomic elements and attributes in staging area need be targeted. This is the priority of the ontological foundation (Wand & Weber, 1990; Wand & Weber, 1999). The interference between services, space, and time aims to build a foundation from variety of combinations of elements and attributes, to cope with interoperability issues such as sequences in building a service dimension. Enterprise integration is involved in tremendous aspects of concepts and technology. Interoperability cannot be achieved by merely consolidating all ad-hoc approaches unless the staging area can be first identified through top-down and bottom-up approaches. It then be represented in three-dimensional form (Green, Rosemann, & Indulska, 2005). In addition, the coherence of three dimensions needs to be built by following the nature of enterprise development. This implies that enterprise integration requires a high-level strategy to envision the macro-scope overview. This gives opportunity for identification of atomic elements to fulfill the micro process design under the macro view (Kalinichenko, 1991). This thesis proposes synthesis of a micro/macro view in maintaining high integration coherence. The coherence is based on an interference approach between disciplines and between dimensions in framework development. In this way the interference can impact space dimension (architecture) and time dimension (agility), and align business value (semantics) though disciplines and mediators.

In order to deal with mediation between virtual and physical elements, this thesis proposes a new concept which can be summarized as follows. Enterprise integration must first be leveraged to the interference of three dimensions: space, time, and value. The space dimension deals with elements, scoping, and scalability. The time dimension concerns the life cycle of the framework and sequence in service aggregation (Hayes, 1996). Finally, the value dimension enables virtual service and life cycle, and envisions the linkage of business values through both macro and micro visions. This concept has convinced industry that the issues of enterprise integration cannot be solved by merely using glued type solutions as it doesn't solve gap and visibility issue within dimensions. Instead, enterprise visions such as semantics homogeneity, and global view

implementation requires a clear visibility of three-dimension approach to address integration gap.

The coherence of the enterprise big picture is shown in Figure 5: for how the virtual and physical elements interact with each other. At the top layer of the figure, enterprise key performance indicator (KPI) and direction identify associated business components for implementation. The identified components are grouped for deconstruction. Meanwhile, the bottom-up approach identifies existing technical assets and physical components for the validation process. So both directions contribute staging area to derive the ontology of atomic elements. Service modeling, which is also related to the synthesis process, evaluates elements for refining and aggregates fine-grain into coarse-grain services. All the metadata regarding validation, analysis, and ontological structure will be stored in repository for alignment, future development, and reuse.

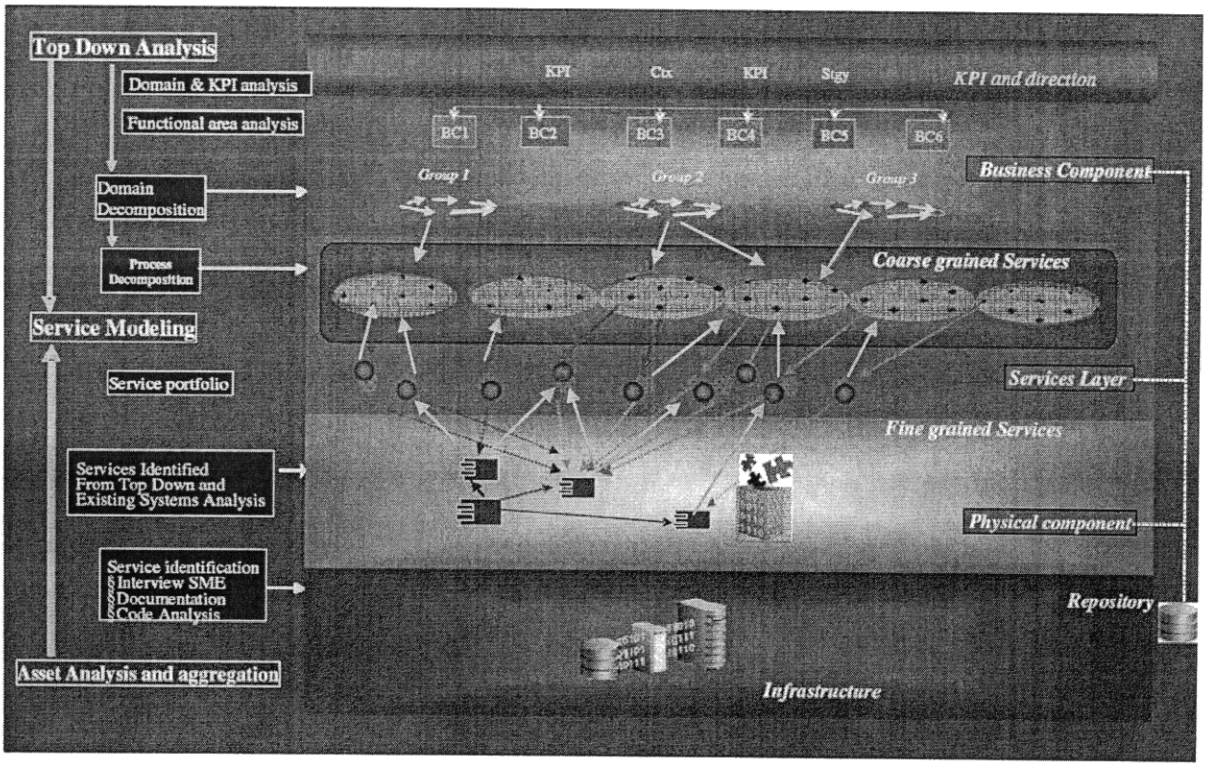


Figure 5. Vertical Coherence in Micro and Macro View

3.1.1 Reason of Ontology and how it is managed

As stated in the first paragraph of this chapter, enterprise integration requires a micro process design under macro view in maintaining a high integration coherence. The following two examples explain how ontology envisions the interoperability between micro and macro view of enterprise integration and how it is managed.

1) Web service modeling ontology (WSMO):

   WSMO is a loosely-coupled approach for web services which uses ontology to maintain a systematic semantics for web service macro architecture. Under this ontology, descriptive semantics is applied to align the ontological properties with a mediator which is developed under micro-view architecture. The ontological approach decouples the service requester from the service provider, in achieving reuse and virtualization objectives.

2) Ontology web language (OWL-S):

   OWL-S defines ontology for services in four different levels: concept, service property, process, and invocation. The hierarchical interference within these four levels constitutes a formal semantic ontology for the choreography of web services.

3.1.2 A Comprehensive Vision of Integration in the Vertical Direction

Individual solutions such as metadata, semantics, patterns and mediation have limited opportunity to single-handedly address root issues. New architecture styles such as SOA, Model Driven (MDA), Event Driven (EDA), and many other emerging technologies have been launched recently (Verma & Sheth, 2007). They aim to encapsulate individual solutions into a dimension-centric approach. However, the lack of a strategy and framework to encompass the realization in semantic mediation, and service synthesis tends to cause weak interoperability coherence. For this reason, this thesis proposes a framework for addressing tactical issues, developed on top of the three-dimensional ontology. From an architecture (space) viewpoint, the synthesis between virtual and physical elements can be standardized for industry compliance. From a sequence (time) viewpoint, an implementation of "state" concept on top of ontological approach allows the micro vision and service transformation to proceed. From a business (value) viewpoint, as the micro vision gives opportunity for semantic development, the service dimension can therefore be built on top of the element ontology with connection to business goals. Once the three-dimensional interference is proceeded under

multi-discipline, the seed skeleton can extend common services horizontally to impact the industry mainstream. It finally achieve service virtualization (Shneiderman & Aris, 2006; Timmerer & Hellwagner, 2005).

3.1.3 The New-Age Service, Integration, and Dimensions

This thesis outlines a distinctive vision, distinguishing itself from conventional approaches. It provides an original knowledge contribution in that it doesn't view integration as merely making two or more objects work together. Most of the literature addresses integration by using a single approach. Examples are message exchange between databases, long process running on different applications, or tiered communication protocols between systems. While these approaches may partially enhance integration, most of them leave tactical issues behind and will never solve the fundamental problems. The potential risk in these approaches is that a pure macro view doesn't address fundamental issues for next-generation computing. Examples are: reuse, alignment, interoperability, agility, and virtualization. Furthermore, the existing literature is not sufficient to encompass enterprise integration from a high-level strategic viewpoint. For this reason, semantics-centric mediation to develop the "vector" and "state" concepts is proposed in this thesis.

3.1.4 The Big Picture of Ontology, Framework, and Strategy

The scope and complexity of enterprise integration are growing with great rapidity. Business innovation and service transformation are changing dramatically, and so must our view of enterprise integration. Parallelism of micro and macro views, top-down and bottom-up approaches, and single source of truth are all basic disciplines for agility that need to be strictly handled (Park & Ram, 2004; Sciore, Siegel & Rosenthal, 1994).

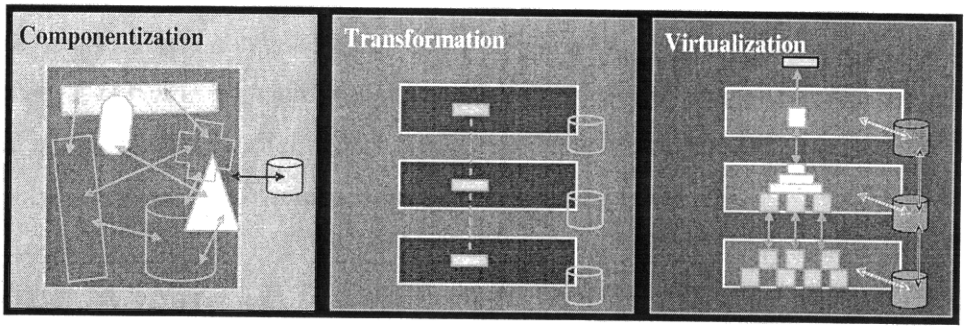


Figure 6. Three-step Approach for Achieving Virtualization

As is shown in Figure 6, the three-step approach which ensures the service dimension and common service is well built. The three-step approach is a part of the META framework which envisions the roadmap instead of architecture. The three-step approach identifies componentization as the first step to deconstruct business and technical aspects and produces atomic elements (Regina et al., 2001). In the transformation step, the ontology of elements will be transformed into a refining process for creating the service dimension (Gruber, 1993). From macro view, synthesis between virtual and physical layers constructs a transparent service layer as mediator. From micro view, the top-down approach and the bottom-up approach construct service components and service dimensions for mediating business and technology. Each service dimension comprises the alignment of candidacies from business element, service component, physical component, coarse-grained service, and semantic metadata. The alignment of dimensions being stored in a repository for choreographic mediation is hidden from users and serves as foundation of service virtualization (Sattler, Geist, & Schallehn, 2005). In the virtualization stage, a macro-view approach moves individual virtual services into common services to form the industry mainstream. Meanwhile, the virtual services contributor will benefit from a common service foundation. On top of this foundation, value-added services will be built.

3.2 Preliminary of the Strategies and Approaches

3.2.1 Common Elements and Aggregation

The determination of common services and aggregated services are a use case driven parsing process. To develop the parsing iteration process, this thesis employs the technique of plotting a CCSA map. Common service is the alignment of a set of business, service, and technical components in such a way that the service dimension is common for reuse. A common component is a component which provisions common service. It should not be limited to a functional element, as a system-driven or information-driven element would qualify as a common component candidate.

An aggregated service is a set of service components, if they follow a particular sequence and dependency with a certain frequency, may qualify for service aggregation. It refines

the mediation between service and business layers from semantics perspective. The objective of service aggregation is to extract an abstract from lower hierachical components to produce a fine/coarse-grain mixed service layer. Two or more atomic service components can be aggregated upward to a coarse-grained service with the threshold of aggregation being determined by the decision-making group. The aggregation process eliminates boundary interconnectivity and alignment complexity, and increases the integrity of common services. Therefore, each coarse-grain service may represent one or more service dimensions, with one side pointed to the business context and the other side pointed to fine-grain services. Together with fine-grain services, fine/coarse-grain services constitute an enriched service layer which facilitates flexible and mature services in mediating business and technology.

3.2.2 Component Synthesis and Interoperability

Synthesis technique is applied in several processes of this thesis in achieving semantic homogeneity. Examples of synthesis application are virtual/physical component synthesis, coarse/fine-grain component synthesis, and semantics-metadata synthesis. Synthesis differentiates itself from alignment or service dimension as synthesis focuses on producing mediators to eliminate conflict and streamline the interference between two or more groups of elements. Rather than being a dimensional approach, synthesis transforms the gap from macro deviation to micro deviation. So the semantic gap can be filled in at micro process which macro process fails to. Furthermore, the main focus of synthesis is to deconstruct the architecture gap to a component level, and applying mediators and semantics to fill in the gap at micro process level. In so doing, the alignment is further tuned into integration coherence through semantic strategy. The philosophy of component synthesis and interoperability proposed by this thesis distinguishes itself from conventional approaches. With micro and macro process synthesis, each atomic element has connection with the business, service, and technical layers in achieving a vertical semantic interoperability. Meanwhile, synthesis balances the integration load by using incremental and iterative approaches, thereby achieving best practice among all optimization approaches.

3.2.3 Semantics, Metadata, and Mediation

As has been indicated, the three forms of repository (semantics, metadata, and service) collaborate with each other to maintain semantic homogeneity. In these repositories are

stored the abstracts of data, structure, alignment, dimension, sequence, and semantic strategy. Therefore the repository plays an important role in maintaining coherence of enterprise interoperability, and also enhances the mediation in both micro and macro view.

3.3 Preliminary of the Framework and Roadmap

3.3.1 Coherence of Micro and Macro View

This thesis proposes micro and macro views to encompass integration coherence. The micro view focuses on interoperability at the element level, such as the component refining process. The macro view focuses on interoperability at the architecture level, an example being the strategy of service mediation between business and technology. Neither approach can be left out in service-based enterprise integration. The interference between micro and macro view such as connectivity of elements to business context, is the key to constructing real enterprise-scale integration. As this aspect is invisible in most of the literature, the approaches so far derived don't effectively impact the coherence and tend to cause bottlenecking. The reasons for this deficiency can be summarized as follows. In a purely micro vision interoperability, the alignment is visible. However, strategies such as reuse and aggregation cannot effectively impact the service layer as the macro vision is missing. Conversely, a purely macro strategy of interoperability only works on interfaces across architecture layers. It doesn't produce a notable value, since micro interoperability is not established.

3.3.2 Interference Models between Framework, Strategy, and Roadmap

The framework is crucial in component-based design from architecture (space), roadmap (time) and strategy (value) perspectives. In the early stage of META framework, the ontology requires taking all heterogeneous designs into interference coherence at component level. It also requires identifying a single source of truth to push service design to be conducted within the framework. The result is that the service components and dimensions are produced in a more standardized and homogeneous way in achieving business goal (Guarino, 1997; Lee, 1996; Ventrone, 1991).

3.4 Four Pillars Concept in deriving Framework

The four pillars in the META framework ensure that the processes address quality of

services in achieving interoperability. This approach facilitates five basic functions: a) scoping, sourcing, and componentization in the vertical direction; b) validation of components in atomic form; c) parsing analysis for identification of common components and service aggregation; d) determining priority and skeleton of enterprise architecture; and e) metadata repository and semantics strategy in moving enterprise interoperability from micro to macro strategy (Mihaila, Raschid, & Tomasic, 2002).

From a service viewpoint, this thesis introduces the concepts of service dimension and of a three-dimensional approach in service interoperability. Service dimension is semantic transmission through metadata design. As dimension is semantic implementation, it gives opportunity for atomic components to be synthesized in micro and macro processes. As the vertical alignment is visible, the semantic-centric metadata forms an enterprise skeleton stretched across the interoperability between business, service, and physical components.

CHAPTER 4

STRATEGIES

4.1 Strategy and Benchmarking for Service-Based Design

By taking concept into strategy. As we have seen, service virtualization is the objective of enterprise integration. To secure this goal, there are two milestones that need be achieved. The first of these is interoperability, which means homogeneity in mediation and vertical alignment. Componentization, component evaluation, service synthesis, and semantic metadata serve as strategies in achieving interoperability (Lewis, 2000). The contiguity of these steps ensures interoperability is established through micro-process analysis, human factor evaluation, synthesis in creating services, and alignment of micro/macro views. Common services, the second milestone, take disciplines of interoperability into common practices across industry. These strategies are mostly covered in multi-discipline approach at conceptual level.

4.1.1 SCIS and Service Component Design

Enterprise integration requires a combination of strategies to attain interoperability. Among all these necessary strategies, SCIS, which is related to synthesis process, is key at this stage. SCIS requires two disciplines to serve as foundation. One is the ontology of atomic elements as mentioned, and the other is a map of vertical alignment: KPI-Business-Services-Components (K-B-S-C). Ontology consists of raw materials and doesn't create dimensions by itself. Only through the interference between these two disciplines are service dimensions proceeded. SCIS is important because it can derive a common practice in defining the component interface and specifying content and semantics interference to construct service-based interoperability. By taking this micro view into a macro view, service-based interoperability applies a semantics metadata approach to maintain coherence of service-based architecture. As the strategies fulfill all technical, service, and business requirements, techniques are applied to approach services, and to stretch services to reach the business goal. This thesis further distinguishes itself from existing literature as conventional strategies usually don't cover these sequential steps and goal orientation in constructing interoperability. Most of the literature identifies only one aspect of techniques, service, or context in its strategy.

4.1.2 Strategy of Enterprise Alignment

This thesis applies both top-down and bottom-up approaches to build up enterprise interoperability. As is shown in Figure 7, initially a single source of truth needs to be identified for compliance purposes. In this example, site E is identified from a set of system sites of $\{A, B, C, D, E\}$ as E holds most of enterprise metadata. The componentization process produces an ontology of enterprise elements which will be stored in repository of E for validation. According to the nature of business implementation, once K-B-S-C alignment and vertical interoperability is established, and the seed skeleton is built at site E, common services can be horizontally extended across each site for industry compliance. This is through metadata strategy.

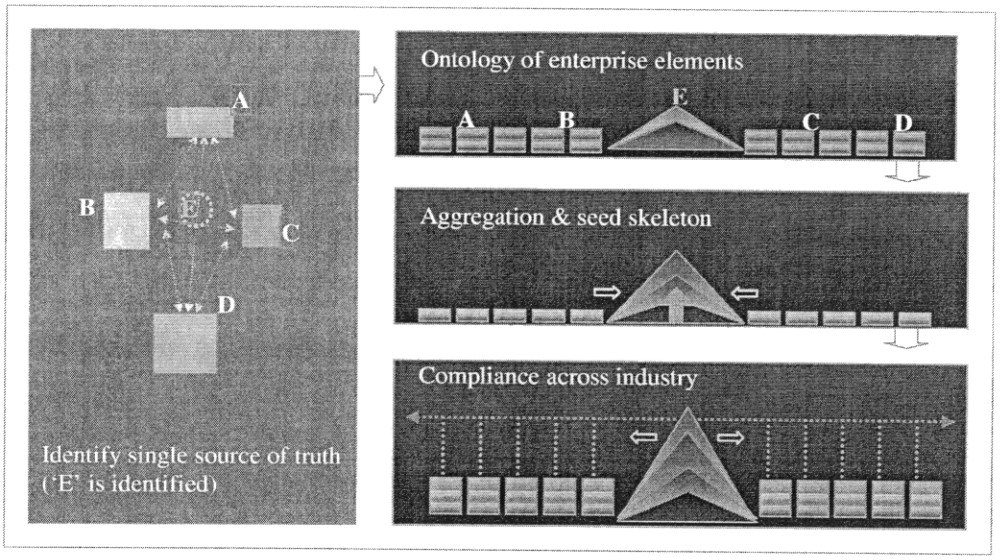


Figure 7. Integration Strategy in Vertical and Horizontal Directions

4.1.3 Strategy of Semantics Interoperability

In the componentization stage, both business deconstruction (top-down) and component aggregation (bottom-up) strategies are employed. Thus semantic abstract is extracted from the large block, into implementation strategy (Wu, Yeh, & Lai, 2006). Meanwhile, the elements derived from componentization form an ontology as indicated in Chapter 3. As the integration covers a mixed ontology of virtual and physical worlds, it is difficult to proceed as different levels of abstract usually cannot be integrated directly. Thus the service layer is identified as a mediating layer into which the abstracts from the business

and technical layers may be taken. The K-B-S-C alignment and refining process produces hierarchical semantic mediators in component form, so they can be designed and refined for reuse. It is important to note that the concept of reuse in this thesis differentiates itself from conventional design. In the proposed hierarchical structure, reuse can be a common component in element form, or it can be an aggregated composite service dimension. Iteration component refining aids interoperability, as does an enriched-service approach. This thesis encompasses loose coupling and a strong vector in semantic alignment. Figure 8 explains the componentization and synthesis initiatives in the early transformation stage. In conventional design, many computing silos such as software code, software program, and data schema are randomly distributed, which may be wire-binding. Following the launching of emerging technologies such as web-based techniques, object oriented programming, and SOA, the element silo will be reworked. It can then be validated as a component, which is the source of ontology of atomic elements. The components, as they are loosely coupled, have opportunity to be mediated to form composite services, also called coarse-grain services. The transition reflects a typical modernization of component-based design and can be leveraged to the early stage of transformation in a three-step approach.

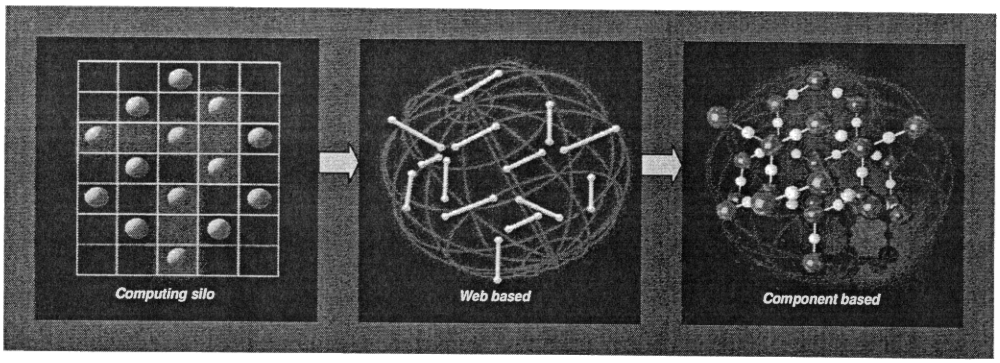


Figure 8. Deconstruction and Synthesis in Early Transformation Stage

When the componentization strategy is taken into the synthesis process, the ontology of elements serves as the foundation of the process. The comprehensive mediation strategy in this thesis is based on technical interfacing, semantic injection, aggregation, and the "vector" concept in service dimension design. Semantic injection produces top-down semantic vector. Together with service patterns, it holds the semantic coherence though service dimension design. It is argued that the interoperability achieved by this approach will be much more solid than that provided by the conventional approaches.

4.2 Metadata Strategy in Component Mediation and Data Aggregation

As has been stated, structure, semantics, and service dimensions are the three key aspects of metadata stored in repository. This thesis handles component alignment by semantic metadata, as it is the primary method for semantic transmission between context, semantics, and policies. From the data aggregation perspective (Madnick & Siegel, 2002; Zhu, Madnick, & Siegel, 2002), semantics play a similar role in mediating service and the component data layer. However, aggregating raw data into a business object may achieve better performance than aggregating service components into a coarse-grain service in some case. It is to be noted that both options produce the same result (Hsu & Rattner, 1993), but the approaches requires benchmarking from related criteria such as performance or implementation effort, to ensure the right approach is applied.

4.3 META Framework Strategy

Ontology of elements, multi-discipline, and multi-dimension are the foundation of the META framework. An ontology is initiated when the componentization process finishes scoping and determines a single source of truth. In this way, each element is produced under the same standard. There are two different options in the formation of ontology. The first is enterprise-based; this option deconstructs the entire business block in a single step and introduces new elements in each succeeding add-on. The second option is project-based; here, elements are added in an incremental way. No matter which option is employed, the result should be the same.

4.3.1 Multiple Disciplines and the Human Factor

The formation of ontology sets up the opportunity for service transformation and virtualization. Technical aspects such as elements specification are already well-addressed in componentization. Therefore, service transformation can apply this specification directly in the synthesis process, and focus the strategy on disciplines and dimensions. Multi-discipline is the strategies, constraints, and principles for developing META framework. It prevents the scenario in which the strategy built for the framework adheres only to one dimension, or being stuck to one narrow area, and is unable to encompass enterprise-level integration. Among the multi-discipline, the human factor plays an important role in componentization as well as semantic strategy. The validation process is

critical from the perspectives of scoping and prioritizing. Before any business change, the decision-making group needs to qualify business components by using a business component map. The goal is not only to ensure that relevant components are selected, but also to determine the components' priority, weight, sequence, and interference. Furthermore, the validation process stores relevant component attributes in metadata repository for evaluation. From human factor perspective, validation process should not merely reflect a single role or isolated thought. The purpose of different weights for different roles is to gather more opinions, from different viewpoints (Boh et al., 2003).

4.3.2 Knowledge Base and Intelligence

Following the validation process is the analysis and evaluation of service-based design. Basically the ontology of elements serves as a knowledge base of raw materials derived from empirical project activities. The concern is now for how much intelligence need be added on top of this knowledge base in order for the pre-built foundation to proceed with automation in service transformation (Pan, 2007). This thesis proposes two approaches. The first is manual configuration for enriched semantic strategy by using empirical metadata. In this way the knowledge base provides sufficient assets and methods for rapid development of a new service dimension. The second approach is an intelligent automated routing for pattern design and component alignment; this approach is most suitable for a large, mature knowledge base. The automation approach helps agility in service virtualization and can be expanded to the industry mainstream if suitable aids such as tools and software can be provided.

4.3.3 Pattern Benchmarking

Service pattern is important in service transformation. Service pattern is the method and format for building service dimensions. Patterns such as top-down, bottom-up, thin service layer, enriched service, object interference model, and vector strategy are all feasible, depending on the business scenario. The strategy in pattern selection is based on the analysis and benchmarking. Basically, an enriched service layer is the pattern for high-complexity architecture, as it requires extensive hierarchy to fill in the gap. It also consists of a mixture of coarse- and fine-grain services to transmit semantics. The object interference model is an integration pattern that works between two or more objects. The conventional n:n pattern has been replaced by hub/spoke 1:n approach, and many enterprise solutions develop a 1:1 solution for run-time purposes. However, the approach

proposed in this thesis aims at a full 1:1 pattern not only for run-time mode, but also from the integration strategy perspective. Vector is a concept which reflects tendency of service for service pattern design. Service components can be produced from either the business or the technical layer. Therefore, each service component can be assessed for its tendency by plotting the vector onto a tendency map for analysis. A higher tendency or in the business direction or coarser grain suggest positioning upper-level services in the service hierarchy. Correspondingly, a higher tendency in the technical direction or fine grain should provision the mediation in the lower levels of the service hierarchy.

4.3.4 Service Layer Mediation Using Hierarchical Abstract

Figure 9 shows the mediation of the service layer between the virtual layer and the physical layer. Service transformation derived from the vertical integration approach produces a hierarchical structure. As we have seen, to integrate the virtual layer and the physical layer need strategy and pattern even when componentization is well established. The enriched service pattern, presented on the right-hand side, facilitate a thick service layer to fill in the gap as the overlap between business and technology is enlarged.

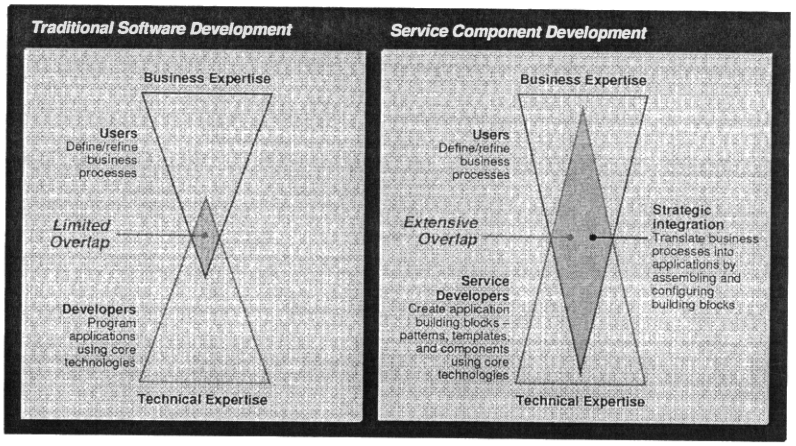


Figure 9. Comparison of Enriched Service Layer and Conventional Design

In this hierarchical structure a component in the upper level works as an abstract of the lower level. The lowest-level service component is coupled directly with one or more physical components. The highest-level service component supports the business context directly for semantic mediation. Together with metadata techniques, this strategy facilitates seamless enterprise interoperability in service transformation.

4.3.5 Strategy in Semantics and Mediators

Kruk, Mocan, & Zaremba (2005) indicated that problems of heterogeneity can be reconsolidated by semantic mediators. Instances of transformations can also be provided based on the similarities identified at the schema level of ontology. However, this thesis points out that, as a semantic strategy is not automatically inherited from metadata, the goal at this stage is to produce the service dimension and semantic-enriched pattern. In order to cope with high complexity and large scale computing, the semantic mediator needs to be implemented on top of the service-based ontology. Instead of merely at a schema level (Nelson & Nelson, 2003), or full automation, semantics need be extended into micro-view service design. The semantic strategy proposed by this thesis achieves interoperability by using some manual mediation techniques for different component forms (Storey, 2005). For this reason, the nature of semantics itself and mediating techniques are all within the scope of this thesis.

There are two areas of semantics that need to be clarified. The first of these is business semantics at the context level (Studer et al., 2006). The second area is semantics content, which is also the information needed in responding to the service request to achieve a particular business semantic (Moulton, Madnick, & Siegel, 2003). Gugliotta et al. (2006) argued that in dealing with interoperability among heterogeneous knowledge sources, a number of mediation requirements need to be considered. These are: a) data mediation at a semantic level, so different concepts can be mapped through mediators; b) goal mediation, so multiple web services can be linked to the same goal via mediators; and c) process mediation, so web service presents a choreography describing how a client talks to the deployed service. As interoperability is the focus in this thesis, we must examine how to mediate these two areas of semantics (context and content) in achieving a business goal. As well, the event trigger technique is also important in supporting dynamic semantic mediation.

4.3.6 Common Component Identification

A common component is a service asset in provisioning common services and the enterprise skeleton. From an enterprise viewpoint, implementation priority, business-critical level, and utilization frequency of a common service are evaluated with a parsing technique. In the ontology, as each component element is labeled, the parsing iteration process reads elements and their movements similarly to an Automata Tuning

machine. It records the "string" it reads, which consists of elements and their dependencies, and produces knowledge-based component analysis. This provides identification of common components in constructing the enterprise skeleton. The seed skeleton is critical in the virtualization stage as it can be viewed as a "superior" coarse-grain service. It plays a critical role from the business perspective as it works as a main route and serves as a mature solution. It further simplifies the interfaces and saves integration effort. For these reasons, the pilot skeleton in constructing the industry mainstream is the most efficient path to achieve service virtualization. From the business benefits perspective, this increases reuse, eliminates integration boundaries, and strengthens enterprise integration.

The final stage is shown in Figure 10: The enterprise transforms all fragmentation and silos into the industry mainstream and common services. As enterprise integration reaches high maturity in this stage, the services, methods, and alignments will become sharable and transparent across industry. This stage aims to form the industry common services following the completion of vertical integration. Horizontal compliance and governance are both important. In addition, semantic metadata and a new concept of velocity to cope with chaotic world are the keys which must be built as foundation of the mainstream.

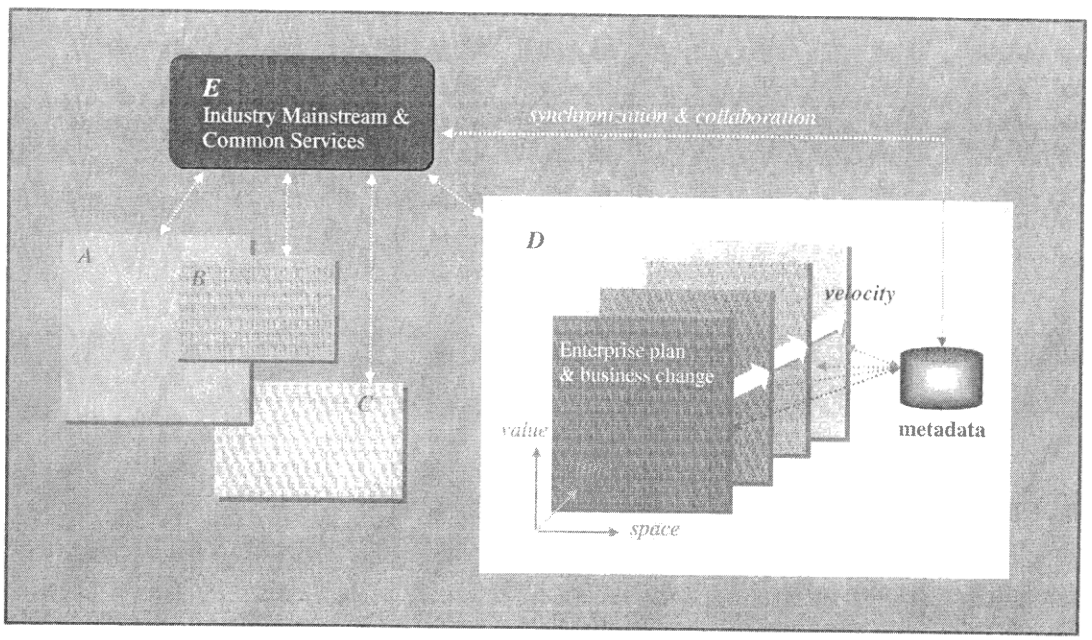


Figure 10. Synchronization between Industry Mainstream and Enterprise Plan

4.3.7 Strategy in Virtualization

This thesis proposes common practices to streamline service virtualization. First, the metadata repository contributes vertical alignment to the enterprise skeleton. It enables the enterprise to stretch out the common practices to increase reuse, to eliminate the integration boundary, and to strengthen enterprise integration. The skeleton also produces metadata interoperability for enterprise integration (Chan, 2006). As all of the organizations share a common practice to communicate their semantic rules, the integration effort can be eliminated. In addition, interoperability can be enhanced once the semantic communication has been fully developed (Morse et al., 2006; Moynihan & Batson, 2006; Wagner et al., 2006).

Service virtualization facilitates industry common services. In this way service consumers share protocol, interface, service components, and information in a semantic homogeneity environment. The hosting and outsourcing services can also be shifted to a collaborative model so that each service provider will have more focus in a specific area. Limitations in organization, metadata, geography, interface, language, and semantics will become transparent (Corcho, 2006). Therefore, service providers can collaborate with other vendors effectively to achieve a mutual objective

CHAPTER 5

FRAMEWORK

Following the concept and strategy from previous chapters, the framework in the realization stage is illustrated in Figure 11. It encapsulates multi-discipline and three-dimension approaches, together with metadata repository and ontology of raw elements, to outline a big picture. It also applies integration principles and techniques to develop the synthesis processes for the governance of service virtualization.

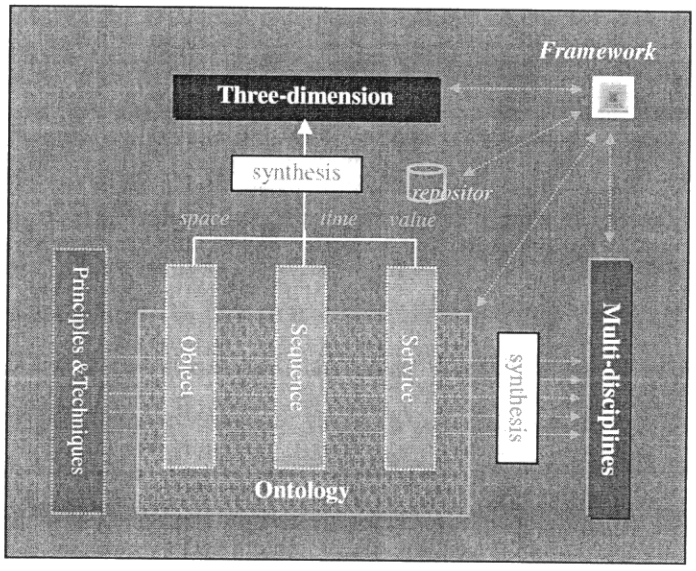


Figure 11. Enterprise Coherence in Constructing Framework

5.1 META Framework Approaches

The META framework aims to streamline the realization of vertical integration, multi-discipline, and three-dimension approaches. The framework consists of a four-pillared (Method, Evaluation, Techniques, and Application) foundation in encompassing strategy. Of these, Method refers to validation methods using group decision-making. Evaluation refers to component analysis and evaluation for producing service dimensions, while Techniques refers to parsing iteration and component synthesis. Finally, Application is virtualization and industry common services. The META framework deals with the development life cycle in a three-step approach (componentization,

transformation, and virtualization). It does this by taking interference between disciplines, dimensions, and ontology of elements into a refining process. The META framework can access visions from different dimensions such as architecture, life cycle, and business values with linkage connected to each other. It requires multi-discipline to ensure the approaches take the strategy and the roadmap into integration coherence as a whole. The process needs many checkpoints at different stages. It also needs many different roles to validate the components, and many patterns to streamline service transformation and virtualization.

5.2 META Framework Interferences

As has been pointed out, the integration scope is so huge that a framework is required to ensure that consensus, compliance, and standards are properly addressed. As a matter of fact, the interference of three dimensions can constitute their focus for service design. Consequently, the interference between disciplines and dimensions can form a matrix to map service design into the framework. As illustrated in Figure 12, the META framework has been developed to streamline service synthesis, and enterprise interoperability. Therefore, the interference between service refining life cycle and architecture can be mapped to META four pillars for realization. Using parallelism of development between K-B-S-C alignment (space), synthesis life cycle (time), and semantic impact (value), the strategy (dimension/discipline interference) is mapped into repository. In this way the strategy will comply with industry common services to impact enterprise integration.

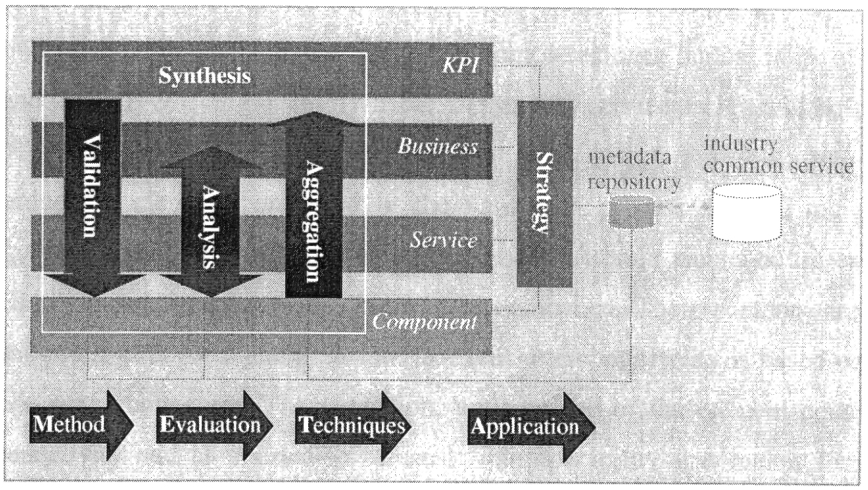


Figure 12. Strategy and Roadmap in Multi-Discipline Framework

5.3 META Pillar 1 - Method and Validation

This thesis applies micro-process synthesis, together with semantic metadata approaches, to derive the strategy in service-based design. The strategy first applies the decision-making process to identify business components for deconstruction. Decision-making and validation in componentization are important as they ensure selected components are exact components which support the business context. The group decision not only supports the component validation process, it produces weighted components for priority reference.

Enterprise vertical integration is a new concept addressing the interoperability between business and IT. Service virtualization is identified as an objective of enterprise integration that requires industry to comply with common services. It also pushes common practices into the industry mainstream (Shafiq et al., 2005). For this reason, the priority should not be the product-based business content or pure technology; rather, strategy is the top priority needing to be implemented. Method is a part of strategy, and the method for component validation is based on the political-model group decision. In group decision-making, each role has a different weight in different criteria. The group members validate business components based on the evaluation of their business significance. The model is then repeated in a use-case deconstruction to derive service components. Same as in business component validation, validation of service and physical components are based on how significant their contribution is to the service dimension which aligned with business goal. The objective of K-B-S-C alignment is to streamline the validation process, and also to build up the vertical enterprise skeleton and extend the key services to their peripheral surroundings. Validation process is completed once the ontology of elements is valid, and the K-B-S-C alignment is visible. In this way service design will be achievable.

The component elements produced in the validation process will be merged into an ontology of elements and become enterprise assets. Atomic elements are raw elements which don't provide any business context or semantics. Their contents are in a more technical and syntax form and their interface and interconnectivity is based on a standard of component architecture. The mediation, built on top of the element content, aims at interconnectivity and at a strategic semantic interoperability that makes loose coupling possible.

5.4 META Pillar 2 – Evaluation and Analysis

Following element validation, evaluation and analysis are introduced to facilitate refining processes in developing service dimensions through synthesis. The key synthesis processes in this stage are: a) coarse/fine-grain synthesis; b) micro/macro synthesis; c) synthesis for semantic/metadata coherence; d) mediation for virtual/physical component; and e) synthesis strategy for common service (Gal et al., 2005).

Synthesis process is the key here, as this stage targets component evaluation for refining. The target for refining is based on the ontology derived from the validation stage. It was stated in the literature review that a service component is an asset with one side pointed towards business and the other side pointed towards technology. Therefore the level of semantics abstract should not be equally distributed across the service layer. Instead, a hierarchical distribution of semantics from low to high in an upward direction is indicated. For this reason, the aggregation process fulfills hierarchical semantics distribution by producing upward abstract aggregation, and extends to the business layer to fill in the semantic gap.

The concept of semantic mediation is based on micro and macro process synthesis. While mediation focuses on micro process, it relies on macro processes such as metadata in maintaining architectural coherence. In addition, mediators such as common components play choreographic mediation roles. They encapsulate both virtual and physical elements in provisioning a service. Besides, the mediators can be functional services or a database elements. As they are frequently used in the enterprise, their metadata is captured and stored in repository either for reuse or for further development. As these common components play a key role in service-based architecture, the determination must be cautious. For the same reason, the parsing iteration process must be strictly executed.

Service aggregation is a composite service aggregated from fine-grained elements. It is validated when the parsing statistics show the sequence and frequency qualify it as an aggregated service. Aggregation is also a key in synthesis process, since it extracts the semantic abstract from fine grains to form a coarser grain. Hierarchical distribution of semantics between virtual and physical layers can be established in addressing semantic gap in enterprise integration.

5.5 META Pillar 3 – Techniques and Design Principle

META framework requires techniques and design to establish micro/macro architecture design. The key techniques and design applied in the framework are as follows. First, in the validation stage, Group Decision Making (GDM) and Political Model Driven (PMD) are used to construct the ontology of elements. Parsing Iteration Process (PIP), Service-Based Pattern Modeling (SBPM), and CCSA processes are used in component analysis and evaluation. Tuning Machine (TM) automata technique is applied to explain the "state" concept. In addition, Service Component Architecture (SCA) is introduced for the service component design in service orientation, particularly in the case of SOA.

5.5.1 META Techniques

To support the realization of META framework, special techniques are applied.

1) Parsing iteration process: A parsing technique is applied to derive statistics for common components and sequence for aggregation services.

2) CCSA process: Derived from parsing techniques, the CCSA process plots the statistics onto a map for synthesis of coarse/fine-grain services.

3) Automata theory: Automata is applied to demonstrate the "state" concept. The concept of service transformation applies an event trigger to leverage the reading behavior of the automata tuning machine. It uses service aggregation of composite services to represent the language acceptable by the machine.

4) Service Component Architecture (SCA): SCA deals with micro-view component architecture design and interconnectivity for the micro-component mechanism.

5) Pattern: Pattern is the strategy of service-based design. A better-tuned pattern guarantees achieving service virtualization as its interoperability is strengthened through synthesis iteration and mediation of semantics.

6) Repository: Repository is the storage of metadata. As has been pointed out, metadata can be in structure, semantics, and dimension form. Metadata in these forms, together with their interferences, construct the coherence of virtual and physical aspects on top of the element ontology. This is the foundation of the strategy.

5.5.2 META Design Principle

Semantic interoperability is a comprehensive service strategy which is critical in achieving common services. From a service design perspective, as service dimension is virtual, it requires mediation techniques to take semantics and metadata into element integration. From design principle perspective, pattern is a symbolic combination of components constructed by following a particular rule or principle. Thus, pattern is the realization of the service dimension. Semantics for pattern design consists of three levels of derivation: technical level, syntax level, and context level. All of these three levels of semantics must be combined and none of them can be missing.

However, semantic interoperability requires a supplementary design when it comes to event triggers for long processes. Event-Driven Architecture (EDA) may be coupled with service orientation to obtain a particularly appealing architecture. It supports the effective execution of possibly distributed and completely decoupled services which can be either synchronously or asynchronously executed. These factors combine to create a global common practice, and for this reason global-local architecture and common practices have become the trend in the industry.

Grain size for interoperability also requires a design principle to proceed component refining process. The business block is huge before componentization, and the original block doesn't commit interoperability because the component alignment is invisible and semantic injection is invalid. After deconstruction, all fine-grain services are in atomic form. The coarser grain formed from fine grains represents a higher degree of abstract of semantics which minimize semantic gap however, the disadvantages is they may loss of flexibility in the new configuration. For this reason, principle of grain size and their ratio for producing a fine/coarse-grain mixed service layer is based on business scenarios and complexity. This factor need to collaborate with strategy of semantics and metadata.

5.6 META Pillar 4 – Application and Virtualization

The nature of enterprise vertical integration proposed by this thesis distinguishes itself from traditional integrations. The proposed approach is not merely the "architecture layer." Instead it consists of a synthesis strategy and framework, to transform the business context and semantics into service dimension toward virtualization. This thesis derives the following disciplines for service design.

1) From "space" perspective: In order to comply with the disciplines, global transformation must be sped up to cope with the semantic gap and misalignment of current non-standard services.

2) From "time" perspective: This thesis develops the foundation and enables capability to cope with agility and velocity instead of adds-on solutions. For this reason, an enterprise coherence of disciplines and dimensions plays a governance role.

3) From "value" perspective: Enterprise integration should be leveraged to business motivation and context as initiatives. The concept of business values should be established to macro vision, and micro implementation as well.

4) This thesis doesn't focus on product-based implementation. Among all disciplines, method should be the priority instead of plug-and-play packaging in order to cope with a chaotic world.

5) Both human elements and computer programs must be able to be comprehended and implemented unambiguously in the realization of enterprise integration.

6) To achieve service transformation, many existing issues need to be settled before the foundation can be built. Typical issues include: a) random integration without disciplines; b) too many integration points and changing frequency; c) data fragmentation making aggregation impossible; and d) invisibility of priorities and common areas.

7) Service transformation also requires industry to consolidate applications in a common foundation to form the mainstream of common practices. Therefore, the service terminology and concepts require a common foundation to leverage each other. Examples are related to aspects such as customer-centricity, master data, single customer view, service insight, and service channel consolidation.

To achieve the goal of service virtualization, this thesis terms the enterprise skeleton the "main route" from the architecture perspective. It stands for a highway for enterprise architecture and provides many service interfaces and patterns in virtual form to eliminate boundaries and leverage virtualization. For this reason the enterprise skeleton needs to be the seed and pilot of service virtualization. Therefore, service consumers can

consume a service at any time, anywhere and by any allowable method. This is the concept of service virtualization, and there are three architecture tasks to reaching the realization of service virtualization which need to interfere with three-step approach:

Task 1. To identify the unique source of truth and consolidate all elements into an ontology leveraged to the source of truth. A repository is used to store metadata for holding enterprise interoperability.

Task 2. To take the vertical integration concept and ontology of elements into an integration coherence to create service dimensions. Metadata and semantics are then applied, through the principles and techniques, to mediate service components and streamline synthesis processes.

Task 3. To develop a framework to encompass methodology, strategy, and techniques to ensure three-step approach is streamlined. Deploy the outcomes from step 2, and use framework to drive the industry key vendors to form an industry mainstream.

The strategy proposed in this thesis has been already used in many successful applications. A notable scenario was an insurance project in 2005 called Core System re-engineering. This was a three-year project in Greater China consisting of 60 IT professionals. My role was that of chief architect and leader of architecture team in charge of the whole enterprise architecture and integration management. The challenge of the project was a huge system with serious fragmentation from its historical burden. The architecture team consisted of 14 leaders. These included 2 leaders for each of three business lines (3×2), system (2), information (2), Quality Assurance (QA) (1), program officers (2), and lead tester (1). It was not easy to reach a consensus initially; however, this was achieved at the architecture stage. At that point DCRP (deconstruct, compose, repository, provision) project methodology was launched in order to build the architecture skeleton. DCRP was the early stage of META framework; it produced the main route of end-to-end processes to envision vertical alignment for fine/coarse-grain synthesis. The skeleton was initially constructed of 67 common components of those business-critical, frequent-access, and common elements to develop service interfaces. The approach contributed significantly to agility to eliminate cost and risk. It was therefore highly recognized by A. Cameron (Australia) and US headquarter authorities,

and was approved for regional certified practice.

Currently, much integration is still lacking a single source of truth and semantic strategy to drive the transformation. The approaches this thesis proposes are intended to address these issues. In addition, virtualization requires intelligence as well as a knowledge base in service dimension design, which is also a pre-requirement for service virtualization.

CHAPTER 6

SUMMARY

6.1 Summary of This Thesis

The unique characteristics of this thesis contribute both in width overview and in depth insight into enterprise integration. It proposes new concepts and takes iteration approaches to derive strategy; it also develops a framework and pushes strategies into realization. This thesis moves through several checkpoints which are summarized as follows.

- Interactive Model: A model which is interactive between research and projects
  A cross-check interactive model between academia and industry is used in this thesis in order to create a close linkage between research and applications. A tremendous amount of project experience contributes initiatives, experiments, and deployment use cases. It also enhances the feasibility and realization of the research this thesis conducts.
- Top-Down Iteration: Iteration approach from concept, to strategy, to realization
  This thesis indicates that an ontology of raw elements should first be constructed, and then the concept taken into strategy to develop a META framework. The iteration approach guarantees that components are produced under uniform processes. Service dimensions are provisioned under consistent approaches. Furthermore, enterprise interoperability is enabled under a single enterprise vision (Barrett, Kaplan, & Wileden, 1996).
- Focus for New Computing: The new concept concerns governance and velocity
  Integrity and responsiveness are the main focus of this thesis. Instead of patching solutions, the integrity and speed with which technologies can approach business change will lead industry into new-generation computing. The new concept indicates that both agility of pre-built portfolios and visibility of enterprise interoperability are foundations of enterprise integration. Additionally, service transformation enables capability in dealing with velocity (Jardim-Gonçalves & Steiger-Garção, 2002).
- Enterprise Vertical Integration: Alignment of business, service, and components
  By following the nature of business implementation, component interference and architecture-layer alignment should take a vertical direction as an approach for

refining iteration. As semantics and service synthesis are initiated by the business context, service mediation and interoperability are also vertical approaches in addressing enterprise vision and reuse objectives. The unique characteristics of vertical integration are recognized as the best practice in achieving service interoperability and common services in many industry projects.

- Ontological Anatomy Micro View: A three-dimensional approach
  - ✧ Space – typical ones are, micro/macro view and fine/coarse-grain components
  - ✧ Time – typical ones are, three-step roadmap and sequence in synthesis process
  - ✧ Values – semantics interoperability and service mediation in creating linkage between business motivation and service dimensions are typical examples
- Integration Coherence Macro View: A multi-discipline approach
  - ✧ Ontology – raw elements in atomic form are deconstructed in componentization
  - ✧ Synthesis – refining iteration in producing service dimensions
  - ✧ Interoperability – enterprise coherence established by micro-view approaches
  - ✧ Common services – vertical integration is stretched out across industry practices
- Virtual/Physical Mediation: Mediation enablement between human and machine

  The vertical integration concept concerns how virtual context and semantics are bridged to the physical components, which is the strategy this thesis proposes (Mikroyannidis & Theodoulidis, 2007).
- Enterprise Interoperability

  This thesis concerns semantics interoperability from both micro and macro views, and mediation between virtual and physical layers. It develops the synthesis process to aggregate fine-grain into coarse-grain services, the objective being to minimize the gap between business and IT.
- META Framework
  - ✧ Method of validation: group decision, and building ontology of raw elements
  - ✧ Evaluation of CCSA and service dimensions
  - ✧ Techniques in parsing techniques and service dimensions design
  - ✧ Applications in industry virtualization
- Industry Common Practices

  Industry urgently seeks a mainstream to push fragmentation, heterogeneity, and silos into common practices. For this reason, this thesis takes vertical approaches into enterprise interoperability, and stretches out the vertical alignment into horizontal compliance across industry.

6.2 Key Contributions

This thesis achieves innovation from many aspects, the key contributions can be summarized as followed:

1) A macro view to envision enterprise vertical alignment:
   This thesis introduces a macro view of enterprise integration by using vertical alignment across business, service, and components. Once this alignment is established, the impact between business and technology is visible and service mediation between the virtual and physical worlds becomes possible.

2) A micro view of component synthesis process:
   Following vertical alignment, the micro view envisions refining at an element level, and aims to apply synthesis process in creating services and interoperability. It follows the nature of business to streamline the integration life cycle. The cost and risk can be minimized by adopting the synthesis process.

3) Industry skeleton and compliance:
   The micro-view approach moves forward to encapsulate the empirical method and patterns. Together with service dimensions, it encompasses business context and produces common services and coarse-grain services. The skeleton becomes transparent for industry compliance and establishes a foundation for industry common practices and service virtualization.

4) Cost reduction by implementing new pattern modeling:
   A three-step approach increases the flexibility in service pattern deployment, particularly for highly complex projects. With this approach, the enterprise doesn't need to stick to one service pattern, instead they can benchmark and select the optimized pattern based on the business scenario. A well-tuned pattern which, for instance, combines aggregated services and common services into the enterprise skeleton can save significant cost in enterprise integration and eliminate project risk.

5) Quality improvement by deploying new techniques:
   This thesis introduces many techniques in service pattern and component design. Technologies such as parsing techniques, CCSA, repository governance, and pattern benchmarking are all necessary in increasing maturity level and quality assurance. Just as a framework takes concept and strategy into realization, a good tool can support a technique and automate the process in achieving quality improvement.

6.3 Limitations

There are three major factors which limit this thesis from a full scope of enterprise integration. These factors are relevant to enterprise integration; however, due to the nature of research fields, they should not be included in this thesis and need separate research.

1) Business innovation is a separate field which affects enterprise integration, as a planned domain development helps compliance of the industry mainstream (Stegwee, 2003).

2) This thesis takes a top-down iteration approach to encompass high-level enterprise integration. Product-based integration should not be the focus of this thesis, however, the gap between "as is" and "to be" needs to be filled in. Before the proposed framework can be effectively pushed into practices, this factor should be addressed.

3) Following the strategy, this thesis suggests implementation of tools and technical details to support the realization of this thesis. Examples include tools for group decision-making, applications for parsing techniques, and automation for synthesis process. The tools are important as they help the comprehension and realization.

6.4 Future Work

This thesis outlines a very "hot topic" and cutting-edge technologies in the industry. Therefore, further research and deployment will be in high demand, and will help industry to build up a worldwide standard and common foundation. The future work stemming from this thesis can be identified as follows:

1) Extensive research and insight are required to shed light on the realization of the META framework. The alignment of Greater China with the world mainstream will generate significant values from a global economy viewpoint. Meanwhile, my interest in teaching and research will produce a collaborative working model. It will also give numerous university students an early exposure to this significant field.

2) To address item 3 in 6.2 (Limitations), initiatives of further research have been planned for the next five years. IBM is a company values innovation. Therefore many engineers have been engaged to investigate the automation and tooling supports of

the approaches this thesis proposes. Tooling requirements such as parsing techniques, group decision, service dimensions alignment, and repository development in holding the ontology of elements structure (such as: IBM WSRR v 6.0, Web Service Registry & Repository) are all typical examples.

3) The transition from current-stage to new-generation computing requires extensive research in the near future as this thesis advises against a "big bang." A period of transition will be involved in standardization and transformation of the industry mainstream. This requires key vendors to collaborate on the methodology and key components to produce a skeleton aligned with enterprise strategy.

4) A further benchmarking of all alternatives is required in future work, to decide on best practice. Benchmarking criteria can be agility, performance, and integrity.

5) Due to cultural differences, each region has a different focus and interpretation within enterprise integration. In one region, for instance, architecture framework is critical for industry compliance. However in another, industry may focus on technical-driven approaches for enterprise solutions. For this reason, continuity of research and publication from what this thesis has achieved are important. Global compliance relies on connective activities such as international forums and conferences for the exchange of key messages. Meanwhile, the interactive model indicated in section 1.2.1 needs to be developed in rapidly developing regions such as Greater China. Governance of creativity is the most effective way of producing power and order.

6) The IT industry requires different intelligence from different cultural backgrounds, therefore "global integration" will be one of the major topics in my future research. Important elements of this future research will be creating linkage of IT with management, philosophy, and arts, and extending the vision of "global integration."

Appendix 1 Statement on Jointly Authored Contributions

One published, and one accepted paper are joint papers by myself and my advisor, Dr. Jie Lu. My advisor contributed extensive discussion and mentoring regarding the structure and writing skills to enhance the readability and expression in a more logical way. These papers may be identified as follows.

- Wu, R., and Lu, J. 2008, 'The Validation Process and Component Analysis in Enterprise Integration', *International Journal of Business and Systems Research (IJBSR)*, vol.2, issue 1

- Wu, R. & Lu, J. 2006, 'Enterprise Integration Strategy of Interoperability', IEEE/CISSE online conference, 4-14 December, 2006, SCSS paper no. 56

Appendix 2 References Regarding to My Publications

ID	Paper title	Journal/Conference	Contact Reference
A3	Micro Process Design and Virtualization In service industry	**ACM**/ EATIS	Prof. Amine Berqia Chair, EATIS 2007 Univ. Algarve, Portugal, chair@eatis.org
E5	The validation and analysis of component based metadata integration		
C5	Semantic interoperability and transformation in enterprise integration	**ACM**/ NZ Journal	Prof. Taiyu Lin, Editor-in-chief, ACM_NZ Inf. Systems Department Massey University, New Zealand t.lin@massey.ac.nz
E4	Determination of Common Component Integration		
M5	Decision Process in Component Based Design		
S5	Enterprise Interoperability and Service Virtualization		
A4	Enterprise Integration in eGovernment industry	**EG** Journal/ Inderscience	Prof. Binshan Lin, Chief Editor, EG BellSouth Corporation Professor College of Business Adm., BE321, Louisiana State University Binshan.Lin@lsus.edu
A5	Enterprise Component based interoperability in e-Government	**Haamaha**	Prof. Stefan Trzcieli , Chair, Haamaha Vice Dean of Computing and Mgt Poznan University of Technology, Stefan.Trzcielinski@put.poznan.pl
E2	The Decision and Analysis of Component Based Integration		
T3	Micro Process Design in Component Based Architecture		
A1	Integration strategy and case study in Financial industry	**IEEE**/ AMS	Prof. David Al-Dabass Conference Chair, AMS 2007 School of Computing & Inf. Nottingham Trent University, UK david.al-dabass@ntu.ac.uk
M1	Enterprise Integration Strategy of Interoperability	**IEEE**/ CISSE	Prof. Khaled M. lleithy, Chair, CISSE Assoc. Dean CS & Eng. University of Bridgeport elleithy@bridgeport.edu
C3	Common component evaluation and micro process design	**IEEE CS**/ EISWT	Prof. John E. Edward, Editor-in-Chief of EISWT, jeedward@gmail.com
M2	The determination of common services and aggregation service		
M3	Service component design and automata		
T4	Implementation Priority and dependency In component based design		
E3	Validation process and component analysis in semantic mediation	**IEEE**/ ICIS 2007	Prof. Dale Jarman, Chair, ICIS, Roger Lee & M Univ. Jarman@ cmich.edu
C2	Service Aggregation and Coarse Grained Based Design	**IEEE**/ ICSSSM	Prof. Ou Xiaojia, leader ICSSSM07 UESTC, Chengdu, China ouxiaojia@uestc.edu.cn
M4	Service Aggregation and Coarse Grained Based Design		
C4	Componentization and semantic mediation	**IEEE**/ IECON'07	Prof. Ren C. Luo, General Chair, IECON, IECON2007@ia.ee.ccu.edu.tw
A2	Case study of component Base Integration in the Industry	**IEEE**/ Southeast	Prof. Gang Tao, Technical Conference Chair, Southeast 2007 Dept of E.E., University of Virginia gt9s@ece.Virginia.edu
C1	Semantic Metadata in Enterprise Integration		

S1	Validation and analysis process in Component Based Design		
T5	Micro Process Design and Services Virtualization		
E1	The validation process and component analysis in enterprise integration	IJBSR Journal	. Prof. Jason Chen, Editor-in-Chief of IJBSR, chen@jepson.gonzaga.edu
S2	Enterprise Integration Framework and metadata	IJIIDS Journal	Prof. Jason J. Jung, Editor of IJIIDS Dept. of CIE, Inha University, Korea j2jung@intelligent.pe.kr
S4	Case study of micro component design and industry integration	IJIIM Journal	Prof. Chang-Tseh Hsieh Editor-in-Chief. IJIIM University of Southern Mississippi Chang-tseh.Hsieh@usm.edu
E6	Component Design, Integration and Service Transformation in Financial Industry	IJISCM Journal	Rrof. David C. Chou, Editor-in-Chief of IJISCM, Department of Computer Inf. Systems, Eastern Michigan University david.chou@emich.edu
T1	Semantic Mediation Modeling in the Service Component Design	IJMSO Journal	Prof. Miguel-Angel Sicilia Editor-in-chief of IJMSO CS/Technical School of Informatics University of Alcal, Spain msicilia@uah.es
T2	An industry case study of micro component design and semantic mediation		
C6	Concept of service provisioning and realization	ISKE 2007	Prof. Qiu Xiaoping Intelligent Control and Development Center, Southwest Jiaotong University, Chengdu 610031, P. R. China Tel: 86-28-87600760
S3	Enterprise Integration in Metadata Environment	JOT Journal	Prof. Richard Wiener, Editor-in-Chief JOT, <rsw@runbox.com>
C7	Architecture Rationalization - part I	EDS Corp. Presentation	Michael Conlin, CTO, EDS APAC 61-2-8965-0500
C10	Architecture Rationalization - part II		
M6	Componentization in SOA Development		
A7	Service Virtualization	IBM Corp. Presentation	David Hodge, Arch. Excutive, IBM 65-6418-4096 David Spizak, S/W Executive, IBM 1-416- 490-4721
E6	SOA Matuality Model Evaluation		
C9	Single View of Customer (SVOC)		
S6	DCRP Strategy in SOA Realization		
S8	High Performance System Design		
S11	Architecture Recomm. in Public Sector		
A8	Financial System Architecture	Oracle Corp. Presentation	Sean Chen, Director, Oracle GC 86-10- 6535-6688
A9	Financial System Integration		
A10	Financial System Operation		
S9	Telecom Architecture and eTOM		
T6	Metadata Computing in CRM		
T7	Event Driven and Batch Process		
T8	Database Partition and CLOB		
C8	DB Staging in B2B	Portal JP Presentation	Hisashi Yoshizako, VP, Portal JP 81-3-5473-8295
S7	Impact Analysis of Platform Selection	Vodafone JP Presentation	Philis Thrush, GM, Vodafone JP 81-3-6403-1014
S10	Telecom Billing and Rating Architecture		

Appendix 3 Bibliography

[1] Alfaro, L., & Henzinger, T. A. 2001, 'Interface theories for component-based design', *the First International Workshop on Embedded Software EMSOFT*, Tahoe City, CA, USA, 8-11 October, 2001, pp. 148-165

[2] Al-Masri, E., & Mahmoud, Q. 2007, 'Interoperability among Service Registry Standards', *IEEE Internet Computing*, vol. 5, pp. 74-77

[3] Aslan, G., & McLeod, D 1999, 'Semantic heterogeneity resolution in federated databases by metadata implantation and stepwise evolution', *The VLDB Journal - The International Journal on Very Large Data Bases*, vol. 8, issue 2, pp. 120-132

[4] Barrett, J., Kaplan, A., & Wileden, J. 1996, 'Automated support for seamless interoperability in polylingual software systems', *ACM SIGSOFT Software Engineering*, vol. 21, issue 6, pp. 147-155

[5] Batini, C., Grosso, R., & Longobardi, G. 2006, 'Design of repositories of conceptual schemas for large-scale e-government projects', *Electronic Government - an International Journal*, vol. 3, pp. 306-328

[6] Benjamin, P., Akella, K., Malek, K., & Fernandes, R. 2005, 'Ontologies in simulation: An ontology-driven framework for process-oriented applications', *the 37th conference on Winter simulation* , Orlando, USA, 04-07 December, 2005, pp. 2355-2363

[7] Bergamaschi, S., Castano, S., & Vincini, M. 1999, 'Semantic integration of semistructured and structured data sources', *ACM SIGMOD Record*, vol. 28, issue 1, pp. 54-59

[8] Boh, W., Yellin, D., Dill, B., & Herbsleb, J. 2003, 'Effectively Managing Information Systems Architecture Standards: An Intra-Organization Perspective', *MISQ vol. 4*, pp. 171-187

[9] Botting, R. 2005, 'Toward Formalizing Domain Modeling Semantics in Language Syntax', *IEEE Transactions on Software Engineering*, vol. 10, pp. 911-915

[10] Bright, M., Hurson, A., & Pakzad, S. 1994, 'Automated resolution of semantic heterogeneity in multidatabases', *ACM Transactions on Database Systems (TODS)*, vol. 19, issue 2, pp. 212-253

[11] Bunge, M. 1977, *'Ontology I: The Furniture of the World'*, Boston: D. Reidel Publishing Company

[12] Bunge, M. 1979, *'Ontology II: A World of Systems'*, Boston: D. Reidel Publishing Company

[13] Bush, J., & Frohman, A. 1991, 'Communication in a network organization', *Organizational Dynamics*, vol. 20, no. 2, pp. 23-36

[14] Chan, L. 2006, 'Metadata interoperability: A study of methodology', *D-LIB Magazine*, vol. 12, no. 6. [online], available from *w.w.w: <http://www.white-clouds.com/iclc/cliej/ cl19chan.htm>*

[15] Chen, L., Shadbolt, N., Tao, F., Goble, C., Puleston, C., & Cox, S. 2004, 'Managing semantic metadata for the semantic grid', *Proceedings of Knowledge Grid and Grid Intelligence (KGGI) workshop* [online], available from *w.w.w: <http://eprints. ecs.soton. ac.uk/10077/01/KGGI2004.pdf>*

[16] Corcho, O. 2006, 'Ontology based document annotation: trends and open research problems', *International Journal of Metadata, Semantics and Ontologies (IJMSO)*, 2006 - vol. 1, no.1, pp. 47-57

[17] Curtis, K., Foster, P., & Stentiford, F. 1999, 'Metadata - The Key to Content Management Services', *The third IEEE Meta-Data conference*, Maryland, USA, April 6-7, 1999, [online], available from w.w.w: *<http://www.ee.ucl.ac.uk/~fstentif/ curtis.htm>*

[18] Davis, L., Gamble, R., Payton, J., Jónsdóttir, G., & Underwood, J. 2001, 'A notation for problematic architecture interactions', *9th Int'l Symposium of Software Engineering ESEC-ACM*, Vienna, Austria, 10-14 September, 2001, pp. 132-141

[19] Dsouza, D. 2001, 'Model-driven architecture and integration opportunities and challenges', *ECOOP 2001*, [online], available from *w.w.w: <http://www.catalysis.org/publications/papers/2001-mda-reqs-desmond-6.pdf>*

[20] Dwarkadas, S., Gharachorloo, L., Kontothanassis, D., Scales, M., Scott, L., & Stets, R. 1999, 'Comparative Evaluation of Fine- and Coarse-Grain Software Distributed Shared Memory', *Proceeding of IEEE HPCA*, Orlando, USA, 9-12 January 1999, pp. 260-269

[21] Gal, A., Anaby-Tavor, A., Trombetta, A., & Montesi, D. 2005, 'A framework for modeling and evaluating automatic semantic reconciliation', *The VLDB Journal - The International Journal on Very Large Data Bases*, vol. 14, issue 1, pp. 50-67

[22] Gleghorn, R. 2005, 'Enterprise Application Integration A Manager's Perspective', *IEEE IT Professional*, vol. 12, pp. 17-23

[23] Goble, C., & Roure, D. 2002, 'The Grid: an application of the semantic web', *ACM SIGMOD Record*, vol. 31, issue 4, pp. 65-70

[24] Gosain, S. 2003, 'Realizing The Vision For Web Services: Strategies For Dealing With Imperfect Standards', *MISQ Special Issue 2003*, pp. 10-29

[25] Gramond, E., & Rodger, S. 1999, 'Using JFLAP to interact with theorems in

automata theory', *ACM SIGCSE Technical Symposium*, New Orleans, USA, 24-28 March, 1999, pp. 336-340

[26] Gray, K., Findler, R., & Flatt, M. 2005, 'Fine-grained interoperability through mirrors and contracts', *ACM SIGPLAN OOPSLA*, vol. 40 issue 10, pp. 231-245

[27] Green, P., Rosemann, M., & Indulska M. 2005, 'Ontological Evaluation of Enterprise Systems Interoperability Using ebXML', *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, pp. 713-725

[28] Gruber, T. 1993, 'A Translation Approach to Portable Ontology Specifications', *Knowledge Acquisition,* vol. 5, issue 2, pp. 199-220

[29] Guarino, N. 1997, 'Understanding, building, and using ontologies', *International Journal of Human-Computer Studies*, vol. 46, issue 2, pp. 293-310

[30] Gugliotta, A., Tanasescu, V., Domingue, J., Davies, R., Gutiérrez-Villarías, L., Rowlatt, M., Richardson, M., & Stinčić, S. 2006, 'Benefits and challenges of applying semantic web services in the e-government domain', *Semantics 2006 Conference*, Vienna, Austria, 28-30 November 2006, pp. 1-12

[31] Hadzilakos, T., Halaris, G., Kavouras, M., Kokla, M., Panopoulos, G., Paraschakis, I., Sellis, T., Tsoulos, L., & M. Zervakis 2000, 'Interoperability and Definition of a National Standard for Geospatial Data: The Case of the Hellenic Cadastre', *International Journal of Applied Earth Observations and Geo-information*, vol. 2, no.2, pp. 120-128

[32] Halaschek, C., Aleman-Meza, B., Arpinar, I., & Sheth, A. 2004, 'Discovering and ranking semantic associations over a large RDF metabase', *the 30th VLDB Conference,* Toronto, Canada, 29 August - 3 September 2004, pp. 1317-1320

[33] Handy, C. 1995, 'Trust and the virtual organization', *Harvard Business Review*, vol. 73, no. 3, pp. 40-50

[34] Harth, A. 2004, 'Mediation Services for Semantic Web Data', *IEEE Intelligent Systems*, vol. 5, pp. 66-71

[35] Harvey, F., Kuhn, W., Pundt, H., Bishr, Y., & Riedemann, C. 1999, 'Semantic interoperability: A central issue for sharing geographic information', *Annals of Regional Science, vol. 33*, issue 2, pp. 213-232

[36] Hayes, P. 1996, 'A catalog of temporal theories', *University of Illinois UIUC-BI-AI-96-01, Beckman Institute*, [online], available from w.w.w *<http://www.coginst.uwf.edu/ephayes/ TimeCatalog1.ps and TimeCatalog2.ps>*

[37] Hsieh, C., Yang, H., & Lin, B. 2002, 'Roles of knowledge management in online procurement systems', Industrial Management & Data Systems, 2002 vol. 102, issue 7, pp. 365-370

[38] Hsu, C., & Rattner, L. 1993, 'Metadatabase solutions for enterprise information

integration problems', *ACM SIGMIS Database*, vol. 24, issue 1, pp. 23-35

[39] Hunter, J. 2001, 'MetaNet - A metadata term thesaurus to enable semantic interoperability between metadata domains', *Journal of Digital Information*, vol. 1, no. 8, pp. 234-253

[40] IBM. 2005, 'WebSphere Developer Technical Journal: Building SOA solutions with the Service Component Architecture - part 1', *press release*, 26 October 2005

[41] IBM. 2006, 'Global CEO Survey', *press release*, 12 July 2006

[42] Janssen, M., & Wagenaar, R. 2003, 'Towards a flexible ICT-architecture for multi-channel e-government service provisioning', *36th Hawaii International Conference on IEEE Computer Society System Sciences*, 6-9 January, 2003, pp. 148-153

[43] Jardim-Gonçalves, R., & Steiger-Garção, A. 2002, 'Developing and integrating enterprise components and services: Implicit multilevel modeling in flexible business environments', *Communications of the ACM*, vol. 45, issue 10, pp. 53-57

[44] Joannou, P. 2007, 'Enterprise, Systems, and Software Engineering - The Need for Integration', *IEEE Computer Society*, vol. 5, pp. 103-105

[45] Kajan, E., & Stoimenov. L 2005, 'The semantic e-business vision: Toward an ontology-driven architectural framework for B2B', *Communications of the ACM*, vol. 48, issue 12, pp. 60-66

[46] Kajan, E. 2004, 'The maturity of open systems for B2B', *ACM SIGecom Exchanges*, vol. 5, issue 2, pp. 34-44

[47] Kalinichenko, L. 1991, 'An anatomy of the information resource semantic abstraction', *ACM SIGMOD Record*, vol. 20, issue 4, pp. 21-24

[48] Kashyap, V., & Sheth, A. 1996, 'Semantic and schematic similarities between database objects: A context-based approach', *VLDB Journal*, vol. 5, no. 4, pp. 276-304

[49] Walsh, K., Anglin, P., Beranek, G., & Wrenn, R. 1997, 'Using Vertical Integration to Improve A Case Study', *IEEE CS 30th Hawaii International Conference on System Sciences (HICSS)* vol. 2, Hawaii, USA, 7-10 January 1997, pp. 396-404

[50] Kolsaker, A., & Lee-Kelley, L. 2006, 'Citizen-centric e-government: A critique of the UK model', *Electronic Government - an International Journal*, vol. 3, no. 2, pp. 127-138

[51] Krichel, T., & Warner, S. M. 2001, 'A metadata framework to support scholarly communication', *Dublin Core Conference 2001,* Tokyo, Japan, 22-26 October, 2001, pp. 1-11

[52] Kruchten, P. 1998, 'Modeling component systems with the unified modeling language', *International Workshop on Component-Based Software Engineering 1998* [online] available from w.w.w: *<http://www.sei.cmu.edu/pacc/icse98/papers/p1.pdf>*

[53] Kruk, S., Mocan, A., & Zaremba, B. (2005), 'Building semantic web services infrastructure for digital libraries', ESWC'2005 [online], available from w.w.w: <http://www.marcont.org/ marcont/pdf/eswc2005_swsl2l.pdf>

[54] Lam, W., & Shankararaman V. 2004, 'An Enterprise Integration Methodology', *IEEE IT Professional*, vol. 3, pp. 40-48

[55] Lassila, O. 1998, 'Web metadata: A matter of semantics', *IEEE Internet Computing*, vol. 2, no. 4, pp. 30-37

[56] Layne, K., & Lee, J. 2001, 'Developing fully functional eGovernment - a four stage model', *Government Information Quarterly*, vol. 1, pp. 122-136

[57] Lee, H,. & Lee, J. 1998, 'Analyzing business domain, a methodology and repository system', *IEEE/IEE Hawaii International Conference on System Science*, Hawaii, USA, 10-13 January 1998, pp. 409-419

[58] Lee, J. 1996, *'Integration Information from Disparate Contexts: A Theory of Semantic Interoperability'*, PhD Thesis, MIT Sloan School of Management

[59] Lewis, D. 2000, *'A Framework for the Development of Service Management Systems for the Open Service Market'*, PhD Thesis, University College London

[60] Liu, C. 2005, 'Using e-governmental indicators to build virtual value chain', *Electronic Government - an International Journal*, vol. 3, pp. 277-291

[61] Lu, J., Zhang, G, & Wu, F. 2005, 'Web-based multi-criteria group decision support system with linguistic term processing function', *IEEE Intelligent Informatics Bulletin*, vol. 5, no. 1, pp. 35-43

[62] Lytras, M. 2006, 'The Semantic electronic government: Knowledge management for citizen relationship and new assessment scenarios', *Electronic Government - an International Journal*, vol. 3, no. 1, pp. 5-17

[63] Mackay, D. 1999, 'Semantic integration of environmental models for application to global information systems and decision-making', *ACM SIGMOD Record, vol. 28*, issue 1, pp. 13-19

[64] Madnick, S. E., & Siegel, M. D. 2002, 'Seize the Opportunity: Exploiting Web Aggregation', *MISQ Executive*, vol. 1, issue 1, pp. 35-46

[65] Maedche, A., Motik, B., & Stojanovic, L. 2003, 'Managing multiple and distributed ontologies on the Semantic Web', *The VLDB Journal - The International Journal on Very Large Data Bases*, vol. 12, issue 4, pp. 286-302

[66] Maedche, A., Staab, S., Stojanovic, N., Studer, R., & Sure, Y. 2001, 'Semantic portal - the SEAL approach', *MIT Press Spinning the Semantic Web*, pp. 317-359

[67] Melnik, S. 1999, 'Generic interoperability framework', *Distributed and Parallel Databases Journal* [online], available from w.w.w: *<http://dbpubs.stanford.edu: 8091/diglib/ginf/WD/ginf-overview/>*

[68] Mihaila, G., Raschid, L., & Tomasic, A. 2002, 'Locating and accessing data repositories with WebSemantics', *The VLDB Journal - The International Journal on Very Large Data Bases*, vol. 11, issue 1, pp. 47-57

[69] Mikroyannidis, A., & Theodoulidis, B. 2007, 'Heraclitus: A Framework for Semantic Web Adaptation', *IEEE Internet Computing*, vol. 5, pp. 45-52

[70] Mirhaji, P., Allemang, D., Coyne, R., & Casscells, S. 2007, 'Improving the Public Health Information Network through Semantic Modeling', *IEEE Intelligent Systems*, vol. 5, pp. 13-17

[71] Mishra, A., Ripley, M., & Gupta, A. 2002, 'Using structural analysis to mediate XML semantic interoperability', *MIT Sloan School of Management publication* [online], available from *w.w.w: <http://ideas.repec.org/p/mit/sloanp/713.html>*

[72] Morse, K., Lightner, M., Little, R., Lutz, B., & Scrudder, R. 2006, 'Enabling Simulation Interoperability', *IEEE Computer Society*, vol. 1, pp. 115-117

[73] Moulton, A., Madnick, M., & Siegel, M. 2003, 'Context interchange mediation for semantic interoperability and dynamic integration of autonomous information sources in the fixed income securities industry', *MIT Sloan School of Management publication* [online], available from w.w.w:*<http://papers.ssrn. com/*sol3/papers. cfm?abstract_id=376863>

[74] Moynihan, R., & Batson, R. 2006, 'Development of a matrix methodology for database reengineering and improvement: An e-government case study', *Electronic Government - an International Journal*, vol. 3, no. 2, pp. 190-203

[75] Mutton, P., & Golbeck, J. 2003, 'Visualization of semantic metadata and ontologies', *Seventh Int'l Conference on Information Visualization IEEE*, London, England, 16-18 July, 2003, pp. 300-305

[76] Nelson, K., & Nelson, H. 2003, 'The Need For A Strategic Ontology', *MISQ vol.4*, pp. 361-371

[77] Nord, J., Nord, G., Cormack, S., & Cater-Steel, A. 2007, 'IT culture: its impact on communication and work relationships in business', *International Journal of Intercultural Information Management (IJIIM)*, vol. 1, issue 1, pp. 85-107

[78] O'Brien, W., Issa, R., Shen, Z., & Xie, H. 2004, 'Configurable environments: a

vision for future project information technologies', *Knowledge Management in the Construction Industry: A Socio-technical Perspective*, pp. 339-358.

[79] Oberle, D., Staab, S., & Eberhart, A. 2006, 'Semantic management of distributed web applications', *IEEE Distributed Systems Online* [online], available from w.w.w:<http://dsonline.computer.org/portal/site/dsonline/menuitem.9ed3d9924aeb0dcd82ccc6716bbe36ec/index.jsp?&pName=dso_level1&path=dsonline/2006/05&file=o5001.xml&xsl=article.xsl&>

[80] Orso, M., Harrold1, J., & Rosenblum, D. 2000, 'Component Metadata for Software Engineering Tasks', *2nd International Workshop on Engineering Distributed Objects,* Davis, USA, 2-3 November, 2000, pp. 129-144

[81] Ouksel, A., & Sheth, A. 1999, 'Semantic interoperability in global information systems', *ACM SIGMOD Record*, vol. 28, issue 1, pp. 5-12

[82] Pan, J. 2007, 'A Flexible Ontology Reasoning Architecture for the Semantic Web', *IEEE Transactions on Knowledge and Data Engineering*, vol. 2, pp. 246-260

[83] Papazoglo, M., & Heuvel, W. 2007, 'Service oriented architectures: approaches, technologies and research issues', *The VLDB Journal - The International Journal on Very Large Data Bases*, vol. 16, issue 3, pp. 389-415

[84] Park, J., & Ram, S. Park 2004, 'Information systems interoperability: What lies beneath? ' *ACM Transactions on Information Systems (TOIS)*, vol. 22, issue 4, pp. 595-632

[85] Petried, C. 2006, 'Semantic Attention Management', *IEEE Internet Computing*, vol. 9, pp. 93-96

[86] Petrie, C. 2007, 'No Science without Semantics', *IEEE Internet Computing*, vol. 7, pp. 86-88

[87] Pinto, G., Medeiros, S., Souza, J., Strauch, J., & Marques, C. 2003, 'Spatial data integration in a collaborative design framework', *Communications of the ACM*, vol. 46 issue 3, pp. 86-90

[88] Qin, J., & Hernandez, N. 2004, 'Ontological representation of learning objects: Building interoperable vocabulary and structures', *The Thirteenth International World Wide Web Conference,* New York, USA, 18-22 May, 2004, pp. 348-349

[89] Qin, J., & Prado, J. C. 2006, 'The semantic and syntactic model of metadata', *Série Comunicação da Informacao Digital*, vol. 4, pp. 143-156

[90] Regina M., Braga, M., Mattoso, M., & Werner, C. 2001, 'The use of mediation and ontology technologies for software component information retrieval', *ACM 2001 symposium on Software reusability: putting software reuse in context SSR '01*, 18-20 May 2001, Toronto, Canada, vol. 26, issue 3, pp. 19-28

[91] Rosenthal, A., Seligman, L., & Renner, S. 2004, 'From semantic integration to

semantics management: case studies and a way forward', *ACM SIGMOD Record*, vol. 33, issue 4, pp. 44-50

[92] Rust, G. 1998, 'Metadata: The right approach. An integrated model for descriptive and rights metadata in e-commerce', *D-Lib Magazine 1998*, pp.1-10

[93] Sattler, K., Geist, I., & Schallehn, E. 2005, 'Concept-based querying in mediator systems', *The VLDB Journal - The International Journal on Very Large Data Bases*, vol. 14, issue 1, pp. 97-111

[94] Scha, R., Bod, R., & Sima'an, K. 1999, 'A memory-based model of syntactic analysis: Data-oriented parsing', *Special Issue on Memory-Based Language Processing*, vol. 11, pp. 409-440

[95] Schiavone, G., Sureshchandran, S., & Hardis K. 1997, 'Terrain database interoperability issues in training with distributed interactive simulation', *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 7, issue 3, pp. 332-367

[96] Schmalz, M. 2003, 'EITH – A unifying representation for database schema and application code in enterprise knowledge extraction', *22nd International Conference on Conceptual Modeling*, Chicago, USA, 13-16 October, 2003, pp. 461-475

[97] Sciore, E., Siegel, M., & Rosenthal, A. 1994, 'Using semantic values to facilitate interoperability among heterogeneous information systems', *ACM Transactions on Database Systems*, vol. 19, issue 2, pp. 254-290

[98] Selvage, M., Wolfson, D., Zurek, B., & Kahan, E. 2006, 'Achieve semantic interoperability in a SOA', *IBM Academy, SOA and Web Services Technical Library* [online], available from w.w.w: <http://www-128.ibm.com/developer works/webservices/library/ws-soa-seminterop.html>

[99] Sicilia, M. 2006, 'Metadata, semantics, and ontology: providing meaning to information resources', *International Journal of Metadata, Semantics and Ontologies*, vol. 1, no. 1, pp. 83-86

[100] Sicilia, M., Garca, E., Pags, C., Martnez, J., & Gutierrez, J. 2005, 'Complete Metadata Records in Learning Object Repositories: Some Evidence and Requirements', International Journal of Learning Technology 2005, vol. 1, no.4, pp. 411-424

[101] Shafiq, B., Joshi, J., Bertino, E., & Ghafoor, A 2005, 'Secure Interoperation in a Multidomain Environment Employing RBAC Policies', *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, pp. 1557-1577

[102] Shneiderman, B., & Aris, A. 2006, 'Network Visualization by Semantic Substrates', *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, pp. 733-740

[103] Smith, K., & Obrst, L. 1999, 'Unpacking the semantics of source and usage to perform semantic reconciliation in large-scale information systems', *ACM SIGMOD Record*, vol. 28, issue 1, pp. 26-31

[104] Sowa, J. 2000, 'Ontology, metadata, and semiotics', *The International Conference on Conceptual Structures*, Darmstadt, Germany, 14-18 August, 2000, pp. 55-81

[105] Stegwee, R. 2003, 'Identification of different types of standards for domain-specific interoperability', *MISQ vol.4*, pp. 161-170

[106] Storey, V. 2005, 'Comparing Relationships in Conceptual Modeling: Mapping to Semantic Classifications', *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, pp. 1478-1489

[107] Studer, R., Ankolekar, A., Hitzler, P., & Sure, Y. 2006, 'A Semantic Future for AI', *IEEE Intelligent Systems*, vol. 7, pp. 8-9

[108] Tannenbaum, A. 2001, *'Metadata Solutions: Using Metamodels, Repositories, XML, and Enterprise Portals to Generate Information on Demand'*, Boston, MA, USA , Addison Wesley Professional

[109] Thanassis, H., Marinos, K., & Margarita, K. 2000, 'Interoperability and definition of a national standard for geospatial data', *International Journal of Applied Earth Observation and Geoinformation*, vol. 2, no. 2, pp. 120-128

[110] Timmerer, C., & Hellwagner, H. 2005, 'Interoperable Adaptive Multimedia Communication', *IEEE MultiMedia*, vol. 1, pp.74-79

[111] Tolosana-Calasanz, R., Nogueras-Iso, J., Bejar, R., Muro-Medrano, P., & Zarazaga-Soria, F. 2006, 'Semantic interoperability based on Dublin Core hierarchical one-to-one mappings', *International Journal of Metadata, Semantics and Ontologies (IJMSO)*, vol. 1, no.3, pp. 183-188

[112] Traugott, K. 2006, 'Electronic thesis and dissertation services: semantic interoperability', *E-Thesis Workshop*, Amsterdam, Netherland, 19-20 January 2006 , pp. 1-17

[113] Uschold, M., & Gruninger, M. 2004, 'Ontologies and semantics for seamless connectivity', *ACM SIGMOD Record*, vol. 33, issue 4, pp. 58-64

[114] Valkeapää, O., & Hyvönen, E. 2006, 'Semantic Annotation with Browser-based Annotation Tool', *1st Asian Semantic Web Conference*, September 3-7, 2006, Beijing, China, pp. 1-4

[115] Vckovski, A., Brassel, K., & Schek, H. 1999, 'Ontology-Driven Geographic Information Systems', *INTEROP Second International Conference*, Zurich, Switzerland, vol. 1580, 10-12 March, 1999, pp. 328-344

[116] Ventrone, V. 1991, 'Semantic heterogeneity as a result of domain evolution', *ACM SIGMOD Record*, vol. 20, issue 4, pp. 16-20

[117] Verma, K., & Sheth, A. 2007, 'Semantically Annotating a Web Service', *IEEE Internet Computing*, vol. 3, pp. 83-85

[118] Vernadat, F. 1997, 'Enterprise modelling languages', *ICEIMT 1997*, Turin, Italy, 28-30 October, 1997, pp. 212-224

[119] Vetere, G., & Lenzerini, M. 2005, 'Models for semantic interoperability in service-oriented architectures', *IBM System Journal*, vol. 44, no. 4, pp. 887-903

[120] Wagner, C., Cheung, K., Ip, R., & Bottcher, S. 2006, 'Building semantic webs for e-Government with Wiki technology', *Electronic Government - an International Journal*, vol. 3, no. 1, pp. 36-55.

[121] Wand, Y., & Weber, R. 1999, 'An ontological analysis of the relationship construct in conceptual modeling', *ACM Transactions on Database Systems (TODS)*, vol. 24, issue 4, pp. 494-528

[122] Wand, Y., & Weber, R. 1990, 'An Ontological Analysis of Some Fundamental Information Systems Concepts', *IEEE Transaction on Software Engineering*, vol. 16, issue 11, pp. 1282-1292

[123] Wu, C., Yeh, J., & Lai, Y. 2006, 'Semantic Segment Extraction and Matching for Internet FAQ Retrieval', *IEEE Transactions on Knowledge and Data Engineering*, vol. 7, pp. 930-940

[124] Wu, R. 2007, 'Integration strategy and case study in financial industry', *IEEE AMS Conference*, 27-30 March, 2007, Phuket, Thailand, pp. 384-387

[125] Wu, R. 2007, 'Semantic metadata in enterprise integration', *IEEE Southeast Conference*, Richmond, USA, 22 - 25 March 2007, pp. 233-236

[126] Yao, W. 2005, 'Interoperability: An indispensable propellant of e-China', *International Journal of Services and Standards*, vol. 1, no. 4, pp. 446-452.

[127] Yu, L. 2005, 'Categorization of common coupling in kernel-based software', *43rd ACM Southeast Conference*, Kennesaw, USA, 18-19 March, 2005, pp. 694-706

[128] Zhang, G., & Lu, J. 2005, 'The definition of optimal solution and an extended Kuhn-Tucker approach for fuzzy linear bilevel programming', *IEEE Intelligent Informatics Bulletin*, vol. 6, no. 2, pp. 1-7

[129] Zhu, H. 2005, 'Effective Information Integration and Reutilization: Solutions to Technological Deficiency Legal Uncertainty', *PhD Thesis, MIT Sloan School of Management*

[130] Zhu, H., Madnick, S. E., & Siegel, M. D. 2002, 'The Interplay of Web Aggregation and Regulations', *3rd International Conference on Law and Technology*, Cambridge, MA, USA, 6-7 November, 2002 [online], available at *w.w.w: <http://web.mit.edu/sloan-msa/Papers/4.5.pdf>*

Appendix 4 Publication List

Starting from this page, the publication attachments are sorted by categories shown in the following order:

Publication category	*Page*
✧ **Section of Publication in Concept**	73
✧ **Section of Publication in Strategy**	149
✧ **Section of Publication in Method**	240
✧ **Section of Publication in Evaluation**	292
✧ **Section of Publication in Techniques**	356
✧ **Section of Publication in Application**	439

[Production Note: Reproduced from the original, best quality reproduction.]

Concept



C1 Semantic Metadata in Enterprise Integration

IEEE Southeast

C2 Concept of Component Aggregation in Inf., Service and Business

IEEE ICSSSM

C3 Common component evaluation and micro process design

IEEE EISWT

C4 Componentization and semantic mediation

IEEE IECON'07

C5 Semantic interoperability and transf. in enterprise integration

ACM NZ

C6 Concept of service provisioning and realization

ISKB 2007

C7 Architecture Rationalization - part I

EDS Corporation

C8 DB Staging in B2B

Portal Japan

C9 Single View of Customer (SVOC)

IBM Corporation

C10 Architecture Rationalization - part II

EDS Corporation

C1

C1: Semantic Metadata in Enterprise Integration

Publication: Southeast07 (IEEE) Conference Procedeing
22 -25 March 2007, Richmond, VA, USA (http://www.southeastcon.org/2007)
ISBN: 1-4244-1029-0

Publication Contact: Professor Gang Tao
Technical Conference Chair
Dept. of E.E., University of Virginia
gt9s@ece.Virginia.edu

IEEE

IEEE SoutheastCon 2007

Date: Tue, 23 Jan 2007 13:20:11 -0800

From: gt9s@ece.Virginia.EDU Add to Address Book Add Mobile Alert

To: rwu_academic@yahoo.com

CC: gt9s@ece.Virginia.EDU

Subject: Your SoutheastCon 2007 Submission (Number 100)

Dear Mr. Raymond Wu:

On behalf of the SoutheastCon 2007 Technical Program Committee, I am pleased to inform you that the following submission has been accepted to the 2007 IEEE Southeast Conference:

Semantic metadata in enterprise integration

as a full paper.

Sincerely,

Gang Tao
Technical Conference Co-Chair
IEEE SoutheastCon 2007

Reviewer's Scores

Originality: 4
Technical strength: 3
Presentation: 5

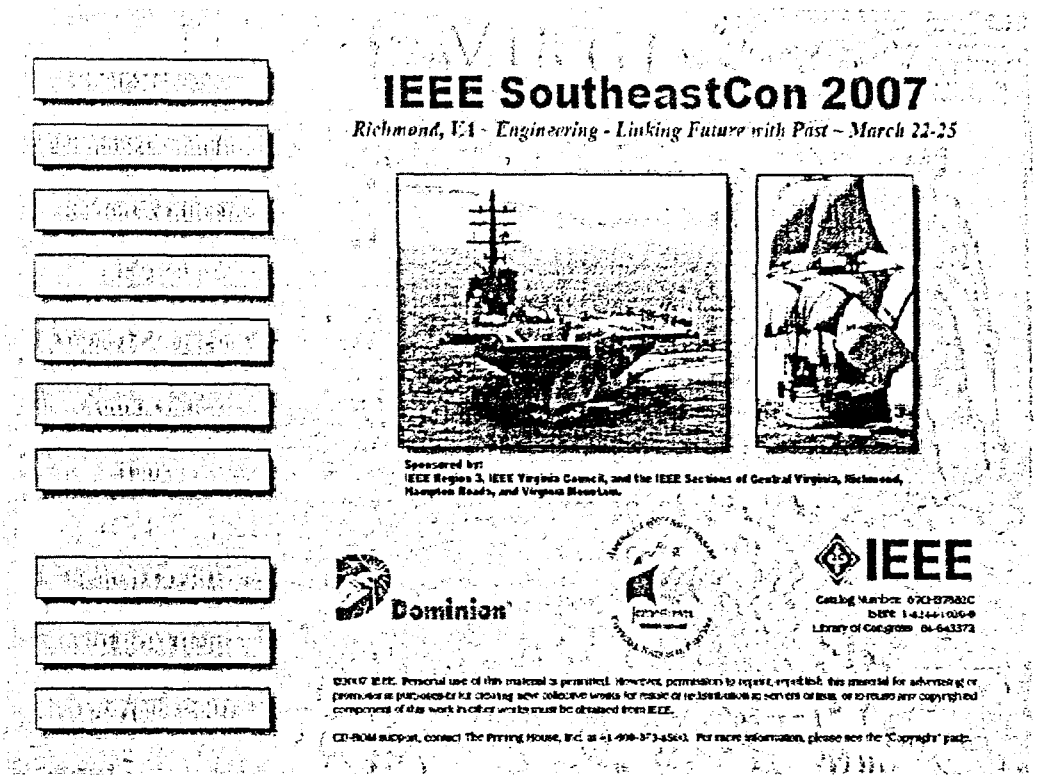


TABLE OF CONTENTS

Scroll to the title and select a Blue link to open a paper. After viewing the paper, use the bookmark "Go to Previous Document" to return to the same page in the Table of Contents.

Computing and Real-Time Systems

Validation and Analysis Process in Component based Design ... 220
[illegible]

Micro Thruster's Design and Services Virtualization ... 225
[illegible]

Designing a pH Data Acquisition and Logging Device using an Inexpensive Microcontroller ... 230
[illegible]

Electric Utility Migration to Regional Transmission Operations: A Case Study ... 234
[illegible]

A New Frontier for Real-Time Systems: Lessons from Molecular Biology ... 239
[illegible]

Software Safety Risk in Legacy Safety-Critical Computer Systems ... 249
[illegible]

Semantic Metadata in Enterprise Integration ... 233
Raymond Wu

An Embedded Linux based Navigation System for an Autonomous Underwater Vehicle ... 238
[illegible]

Case Study of Component base Integration in the Financial Industry ... 243
[illegible]

1-4244-1029-0/07/$25.00 ©2007 IEEE.

Semantic metadata in enterprise integration

Raymond Wu
IBM Software Group
raymond@de.ibm.com

Abstract

Semantic metadata involved in information aggregation in component based service, it allow two separate entities coordinate and ultimately decide on the same implicit ground-rules captures the essence of interoperability. Semantic metadata is crucial for enterprise interoperability to integrate the metadata between business, service and information layers, this create visibility of vertical alignment within enterprise architecture and use metadata configuration to construct the mapping between each layer. There are four basic type of semantic metadata, top down structure retrieval, bottom up information aggregation, remote meta script execution and retrieval and repository in holding the integration interoperability. Each type of metadata play important role and contribute their values to enterprise integration.

Introduction

In our recent research, componentization, metadata and business semantic are all the foundation in enterprise integration, we proposed component based validation, analysis, design and provisioning as our approach and metadata interoperability between business, service and information as our strategy [1]. For this reason, semantic metadata, which mediate business semantic, service and information metadata is crucial in enterprise integration. Metadata computing was followed by the construction of a metadata repository; one of the purposes of the innovation of recursive metadata iteration (self-built scripting and execution) and cursor computing ("join-query execution" within memory) was an intention to build the metadata engine in communication with other repositories in Java/XML/SQL formats. T. Krichel and S. M. Warner designed new metadata format to advance scholarly communication over the Internet, to be used within the Open Archives Initiative. It is based on work by the Dublin Core Metadata Initiative and others and present a requirements analysis and then propose a conceptual framework for the metadata [2].

The other purpose was to use metadata in dynamic iteration, combined with cache memory computing, to produce a high integrity and high performance output in a real-time operational environment. An example of this is that by allocating summarized course information in a metadata repository as the driving force in extracting a student's study plan, the query response time was reduced from several seconds down to one nanosecond [3].

Semantic metadata

A Tannenbaum in her "Metadata Solutions" offers an enlightening overview of the role of metadata within current IT trends, provides detailed treatment of metadata, metamodels, meta-metadata, and meta-metamodel concepts and structures [4].

Firstly, Componentization is important in supporting vertical business layer integration. As we were all aware, traditional silo implementation was aimed at the end-to-end business process, and supported project architecture but also caused many problems of fragmentation in enterprise architecture. Many concepts of service components, common interfaces, shared services, customer insight, single view of data and technical assets were unable to be achieved and reused due to n:n issues and their consequent misalignment; therefore, our first step was to build the single source of truth and metadata aggregation for further development. From bottom up service provisioning perspectives, this was mainly to deconstruct the business by using a "separation of concern" concept and a "componentization" approach to produce the common components and interfaces, this was the initiative of metadata repository design as data fragmentation and duplication need to be removed before any development of data aggregation and metadata design. Therefore our first step was consolidate and identify the master data for further implementation of metadata.

The second steps was the actual work in building metadata structure which was completed by a huge ETL enterprise project (extraction, transformation and loading) and data cleansing processes to identify those randomly distributed data to be aggregated toward the metadata-oriented design, and this is the foundation of building system level of metadata.

The third steps was related to the transaction level of metadata which is supported by the information aggregation concept such as, data partition and business objects so that any record-level transaction will be recorded and logged into a "summary" record reflecting the meta-level business transaction and aggregation information for transaction data.

The fourth step was using summary information, stored in the metadata repository, to provides upward provision of

1-4244-1029-0/07/$25.00 ©2007 IEEE.

summarized business intelligence with linkage to upper layers' repositories for collaboration and vertical interoperability against business changes or add-ons. L. Chan pointed out that as the rapid growth of Internet resources and digital collections and libraries is accompanied by a proliferation of metadata schemas. Each metadata schema has been designed based on the requirements of the particular user community, intended users, type of materials, subject domain, the depth of description, etc. Problems arise when building large digital libraries or repositories with metadata records prepared according to diverse schemas. Most users do not and should not have to know or understand the underlying structure of the digital collection; but in reality, they are experiencing difficulties in resource discovery and access. How to enable a seamless interoperability and search among metadata schemas presents considerable challenges [5]

The fifth and last step was related to semantic metadata maintained in semantic map, the semantic map is a repository of semantic associations between named data elements. The use of the semantic map allows each data user and each data source to maintain its own vocabulary as best suited for its own objectives. The association maintains considerable flexibility in relating different vocabularies and encapsulates the changes needed to capture modifications to vocabularies or data field relationships.

For the simple data access case currently being described, the target and target elements are the search criteria, source, and the associations are the return values. In other uses of the semantic map information, other combinations of entities in this relation can be supplied as search criteria, with the remaining entities comprising the return information. The data element names themselves are not required to convey any semantic content and the semantic map knows no details of any association, other than that one exists. Associations (or pointers to associations) are information inputs to the semantic map, not a part of its structure or implementation. Thus, associations and the associated data sources can be created or modified without requiring modification of the semantic map infrastructure itself. A data access method is an executable corresponding to a data source, which extracts data values from the data source. A data access method understands the means by which data values are stored and/or generated at the data source and can accomplish access of values for a given list of data elements known to the data source. It is assumed that the data access method takes as its arguments a list of data element names and returns values corresponding to those names. The actual processing may be a simple SQL call to a database or it may require significant processing and authentication to satisfy access privileges. In any case, the access is accomplished without any knowledge of how the accessed values will eventually be used, unless this is required as part of the user authorization.

Research has approached the challenge of managing and integrating disparate data resources in many ways in the past. Metadata needs many processes to gather, retrieve, transform, manage and model, and its scope covers metadata format, modeling, processing and integration strategy. L. Chen proposed a generic framework for managing resource SMD, in which ontologies are used for metadata modeling and the Web Ontology Language (OWL) for semantic representation. Generated resource SMD are archived in a knowledge repository enhanced with Description Logic (DL) based reasoning capability. A raft of tools, mechanisms and APIs are developed to support SMD management lifecycle, including metadata generation, semantic annotation, knowledge storage and semantic reuse [6].

Metadata is the aggregation abstract or definition of data; it can be in the form of information driver, data aggregation, a script or a description of the raw data. Metadata computing can be used as an efficient way to synchronize between databases, or symbolic "highly condensed" information for an easy retrieval of identified records. We extended this concept to other areas such as component-based implementation, and business component decomposition. Refer to Figure 1, there are four main groups in achieving information metadata:

a) A bottom-up data aggregation and system abstraction from the infrastructure

The concept of transaction aggregation is supported by data partition and business objects such that any record-level manipulation will be recorded and logged into a "summary" record in reflecting the meta-level business transaction. The meta information, stored in the metadata repository, provides upward provision of summarized business intelligence with linkage to upper layers' repositories for their collaboration and vertical inter-operability against any business changes or add-ons. By storing the partition information in a metadata repository with linkage to service metadata, the system integrity and performance will be improved dramatically. Each pending record can be centralized in 1% of the database for further service until completed. By this means each query hit-ratio is nearly 100% and there is great improvement in SLA (service level agreement) efficiency. Thus interoperability can be attained within the meta-layer to effectively impact the individual entity.

b) A top-down matrix mapping of semantic messages in which the mapping information is stored in a metadata repository

Figure 2. shows Metadata recursive iteration is followed by the construction of the metadata repository; one of the purposes of the innovation of recursive metadata iteration and cursor computing was an intention to build the metadata engine in communication with other repositories formats; the other purpose was using metadata in dynamic iteration, combined with cache memory computing, to produce a high integrity and high performance output in a real-time operational environment.

Information integration on a global scale requires a new approach to achieve a new level of seamless and uniform access to disparate data sources. This requires a server and a

1-4244-1029-0/07/$25.00 ©2007 IEEE.

repository to provide a platform through which an enterprise can manage and provide uniform access to disparate information systems for timely application development in a hyper-competitive inter-networked world.

Desmond D'Souza defines vertical integration in his "MDA (Model-Driven Architecture) and Integration Opportunities and Challenges" and points out that vertical integration came from the different levels of abstraction within the same system, physical data models, logical data models, networks, application specifications, component assemblies, business process models, and business goals and strategies [7]. In each strategic implementation, the interoperability is built by using semantic metadata between the layers by using top-down (from strategy to component) and bottom-up (components provisioning the upper service and business layers) approaches, and shows how the information and services form common components and aggregated components in creating the vertical coherence of enterprise integration. The above mentioned database partitions support the information metadata in providing information metadata in the form of common components or aggregated components in the bottom-up approach.

To overcome limitation in semantic mapping, a metadata term thesaurus which provides the additional semantic knowledge has been developed to enable flexible and dynamic mapping among metadata standards [8].

c) System automatically generate a script and send it to the target database for execution, system remotely execute the run file, spool the result and status and retrieve it through file transfer. The script for remote execution is a kind of metadata as it hold the semantic information and abstract structure for remote database to manipulate its data and query the result into spool file for retrieval [9].

d) Figure 3. illustrates the transformation of service components, aggregation in the service layer is important as it eliminates boundary configuration, increases the integrity of common services in an error-free configuration environment, and furthermore can be developed into coarse grain processes or processes adjacent to the business layer. Therefore the referential alignment of business-service strategy uses coarse grains to support business components directly, for those fine grains at the bottom of the service layer can form a different pattern in supporting the coarse grains. These metadata regarding aggregation are stored in metadata repository, they have capability in constructing skeleton of service virtualization within enterprise integration [10].

The data access service is the central component in mediating and coordinating a data access request. The service (1) takes as input a list of names corresponding to target entities for which values are needed, (2) calls upon the semantic mapping service to identify (one or more) associations to (one or more) data sources and the corresponding data source elements needed to generate these values, (3) retrieves data element values from the data source, (4) invokes the processing of the associations (including passing of data element values retrieved from the data source) to generate values for each name on the target request list, and (5) returns the target values to the entity which invoked the data access service. The data access service will invoke data access methods to extract values from corresponding data sources for use as input to the associations.

Conclusion

Metadata is the concept and definition about data; typically metadata describes the structure and location of data, it describe how pieces of information in the organization relate to each other. We categorize the metadata into four groups, the system level, transaction level, meta script for data retrieval and semantic metadata in holding the integration interoperability. The system metadata related to system configuration and DDL (data definition language) which supported parameterization in metadata computing, while transaction metadata related to business transactions in daily business operation. In master data environment or centric database, all the distributed databases can use self-generated script for a remote master database to run the scripts on a global database as a single source of truth, so each peripheral local database doesn't need to send a high volume of data to manipulate its global database. The only update is small file of either runtime script or identified records with their record key; this saves huge network and database loads. Another advantage of metadata computing is that there is a savings of computing resources: When there is a need for further statistics the system doesn't need to retrieve a huge volume of raw data each time, which may cause locking issues. Semantic metadata holding those common and aggregated services, in the form of aggregated information, can be a good reuse object asset, so several interested fields of information can be aggregated toward metadata for further computing.

1-4244-1029-0/07/$25.00 ©2007 IEEE.

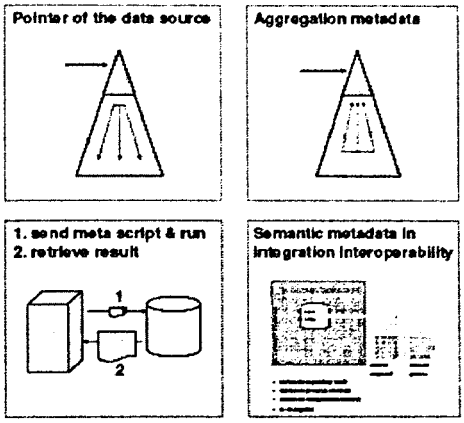


Figure 1. Four types of metadata

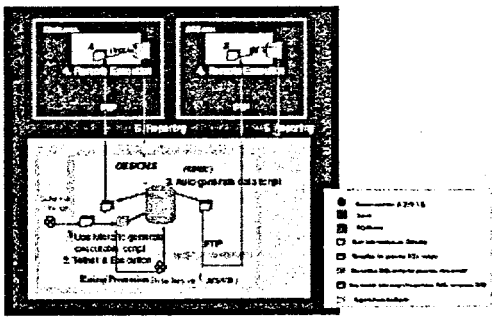


Figure 2. Remote meta script execution and retrieval

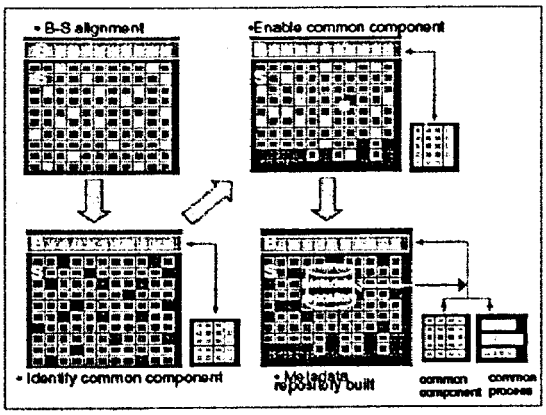


Figure 3. Semantic metadata stored in repository holding integration interoperability

Reference

1. A. Orso1, M. Jean Harrold1, and D. Rosenblum, (2000), "Component Metadata for Software Engineering Tasks", Proceedings of EDO 2000, LNCS Vol. 1999, Springer
2. T. Krichel and S. M. Warner, (2001), "A metadata framework to support scholarly communication", Dublin Core Conference, 2001
3. R. Heery and H. Wagner, (2002), "A Metadata Registry for the Semantic Web", D-Lib Magazine May 2002 Volume 8 Number 5 ISSN 1082-9873
4. A Tannenbaum, (2001), "Metadata Solutions: Using Metamodels, Repositories, XML, and Enterprise Portals to Generate Information on Demand", ISBN-10: 0-201-71976-2: ISBN-13: 978-0-201-71976-5
5. L. Chan, (2005), "Metadata Interoperability: A Study of Methodology", 3rd China-US Library Conference
6. L. Chen and N. Shadbolt and F. Tao and C. Gobleand C. Puleston and S. Cox, (2004), "Managing Semantic Metadata for the Semantic Grid", In Proceedings of Knowledge Grid and Grid Intelligence (KGGI) workshop
7. D. DSouza,(2001)," Model-Driven Architecture and Integration Opportunities and Challenges", Kinetium
8. J. Hunter , (2001), "MetaNet - A Metadata Term Thesaurus to Enable Semantic Interoperability Between Metadata Domains", Journal of Digital Information, Volume 1 Issue 8
9. C. Halaschek and B. Aleman-Meza and I. .Arpinar and A.Sheth, (2004), "Discovering and Ranking Semantic Associations over a Large RDF Metabase", Proceedings of the 30th VLDB Conference, Toronto
10. M. Scott, (1999), "Virtual Organization as Process: Integrating Cognitive and Social Structure Across Time and Space", Journal of Computer-Mediated Communication and Organization Science

1-4244-1029-0/07/$25.00 ©2007 IEEE.

C2: Concept of Component Aggregation in Information, Service and Business

Publication: ICSSSM (IEEE) Proceeding June 2007

The Fourth IEEE Conference Service Systems and Service Management

June 09-11, 2007, Chengdu, P. R. CHINA (http://www.mgmt.uestc.edu.cn/icsssm07)

Publication Contact: Professor Ou Xiaojia

Leader ICSSSM07

UESTC, Chengdu, China

ouxiaojia@uestu.cn

IC SSSM'07

IEEE

Chengdu 610054, P. R. China
2007-3-20

RE: Paper No.A052

TITLE: ※Concept of component aggregation in information, service and business§ by: Raymond, Cheng-Yi Wu

Dear *Raymond, Cheng-Yi Wu,*

On behalf of the ICSSSM2007 Program Committee, we are pleased to inform you that your paper has been accepted for presentation at **The International Conference on Service Systems and Service Management (ICSSSM2007)** being held in Chengdu, China. Congratulations! The acceptance is conditional, upon receipt of the following three items by April 20, 2007.

1. The electronic version of your camera-ready paper in .PDF or .DOC files (see attached *style.doc* for your paper format preparation), the maximum length of the paper is 6 pages and extra page will be over charged. (Please send us by email.)
2. Copyright Transfer Form completed and signed. (Please send us by email, fax or mail.)
3. Completed registration form and fee for each presenting author. (Please fill it out, signed, send us by email, fax or mail.)

Note: Please send us by email to ouxiaojia@uestc.edu.cn.

I would like to take this opportunity to thank you for submitting your paper to the in Chengdu.

Yours sincerely,

Yong Zeng

Yong Zeng
Conference Chair, ICSSSM2007

The Fourth IEEE Conference on
Service Systems and Service Management (ICSSSM2007)

June 09-11, 2007, University of Electronic Science and Technology of China
Chengdu, P. R. CHINA

Professor Yong Zeng

Conference Chair, ICSSSM2007
School of Management
University of Electronic Science and Technology of China
Chengdu 610054, P. R. China

2007-3-20

Paper Review for ICSSSM'07

Title: Concept of component aggregation in information, service and business

Authors: Raymond, Cheng-Yi Wu
Paper Number: A052

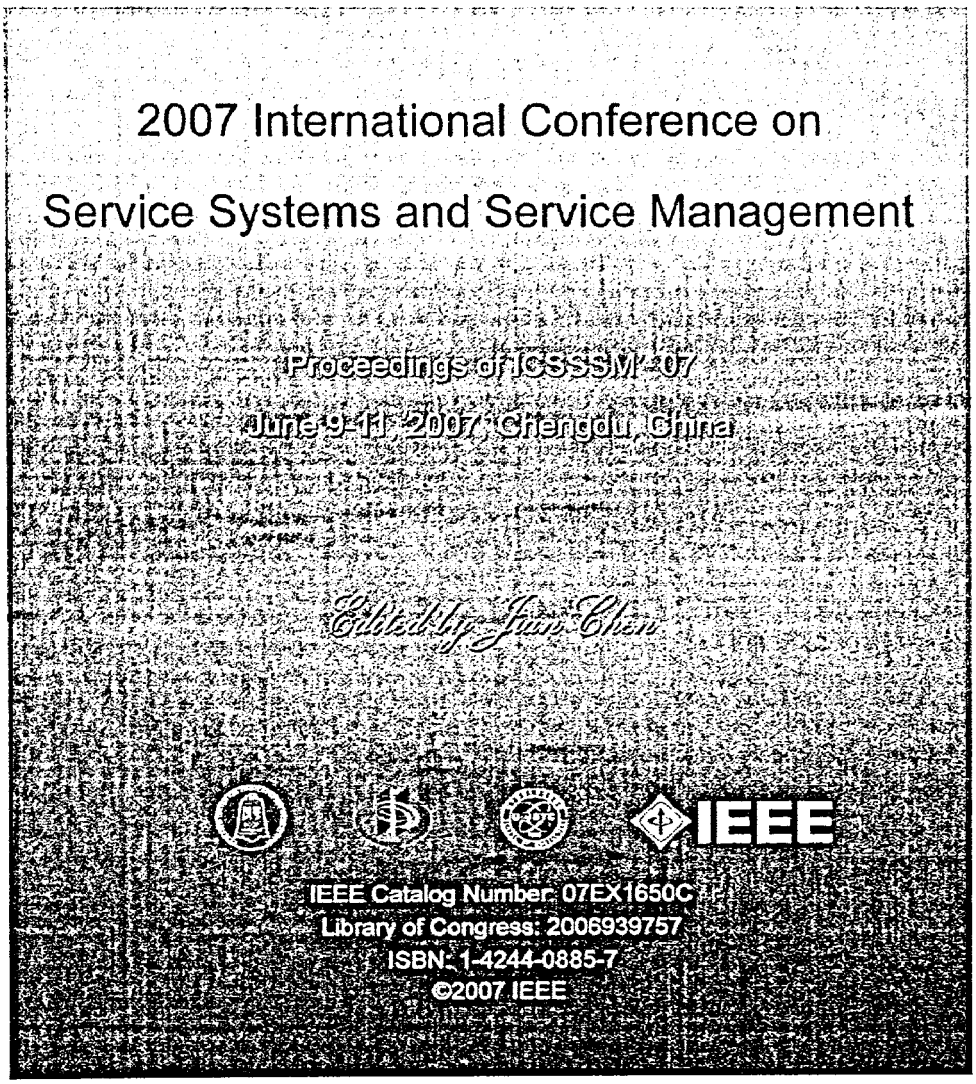
	Excellent	Good	Average	Weak
Bibliography	X			
Contribution	X			
Originality	X			
Practical interest	X			
Clarity		X		
English	X			
Respect of the conference format		X		
Numerical experiments (if necessary)		X		

Dear Raymond Wu ,

On behalf of the ICSSSM2007 Program Committee, we are pleased to inform you that your paper has been published in the proceedings of ICSSSM'07.

ICSSSM07 Organization Committee
Phone: +86-28-83206652
Fax: +86-28-83206652

E-mail: ouxiaojia@uestc.edu.cn
School of Management, UESTC
Chengdu, 610054, China



PART IX Service Information Systems: Design, Implementation and Adoption

A Service-Oriented Conceptual Framework of Software Configuration Management	Ping Liang, Jianfang Li
A High Performance Architecture for Web-Service Systems in XML	Fangru Wang
A Load Balancing Technology Oriented to Common Service Information System	Cuiqing Jiang,Jing Zhang
A Semantic Web Service Oriented Model for E-Commerce	Jing Ni, Xiefi Zhao, Hijun Zhu
A Study on Critical Success Factors in ERP Systems Implementation	Nanzhe Jing, Xun Qiu
BLOGS: A Smart Book Locating System Based on RFID in Library	Kunsheng Sun, Yanlin Li
Concept of Component Aggregation in Information, Service and Business	Raymond Wu
Customer Information System for Product and Service Management	S. Jia Phua, Wee Keong Ng
Towards Knowledge Extraction from Textual and Mixed Format Data	
Haitang Liu, Xiang Li, Bin Song	
Design and Overall Framework of Material Information Support System for GSS	Jia Li, Pengzhu Zhang, Yunhu
Jiang	

Concept of component aggregation in information, service and business

Raymond Wu
IBM Software Group
raymond@de.ibm.com

ABSTRACT

Service aggregation play crucial role in service provisioning. for those components that follow a certain sequence and perform a combination of coarse grained services. they are qualified for service aggregation. Service aggregation is validated in parsing statistics. this pre-built coarse grained service provides upward service provisioning and brings in its values in eliminating the integration interface and risk caused by boundary implementation. In the new age business driven services. coarse grained services construct the foundation of enterprise skeleton and rely on the aggregation from fine grained services. this shorten the gap between business and IT and streamline time to market. The skeleton it produced exploits a loosely coupled approach in creating integration interoperability and constructing the foundation for service

1. INTRODUCTION

Service metadata holds mapping and routing intelligence that is stored in a service repository so that the component aggregation have an opportunity to form the main route of the common service skeleton; any business semantic translation to the process and infrastructure level becomes feasible and efficient under this main route skeleton [1].

Service aggregation means a composite service aggregated from at least two components that follow a certain sequence and perform a combination of coarse grained services. Service aggregation is validated when the parsing statistics show a combination of services following a unique sequence frequently forming either an aggregated service or potential service-based process; the pre-built coarse grained service provides upward service provisioning and brings in its values in eliminating the integration interface and risk caused by boundary implementation [2]. Once the visibility of vertical enterprise alignment is created, the metadata of a component-based configuration forms a clear picture of the enterprise skeleton using metadata integration between business. service and information. Through a metadata approach. we can either reuse the metadata mapping and configuration from successful experience or construct a new map for the project by using the configuration of common components and service aggregation as a project portfolio while extending the implementation to configuration of less significant components. Moreover, once the service components and aggregation services are built they can either be reused or aggregated to a coarser grain later if the aggregation is common and optimized in the parsing evaluation. These processes produce metadata interoperability for enterprise integration.

The information aggregation in the component layer forms the metadata that is stored in an information repository distinct from the service components. It constructs part of the component element in the physical layer. The component element consists of physical units of functional and non-functional elements; while an agent is an example of a functional element, a business object or a security component are both considered as non-functional elements.

Service aggregation is a process for deriving the metadata within a layer, so even though the service components and component elements have been well defined in the previous 1-5 steps. fine tuning is required to derive the final step.

Within the component layer there is no such process; as the physical layer is at an atomic level loosely coupled with peer components and associated services. we assume all the component elements in information component form are in an optimized state and that no further aggregation is required. Therefore, before any information component is defined, a suitable aggregation in optimization is necessary; this implies that the aligned service component is always associated with an optimized component. Thus, if the component is an information component, there is no further aggregation required after S-C is configured. Assume the value we derived from the political model $\Phi^*(SC^k_i)$ satisfies the eigenvalues approach and works as an eigenvalue of the rational model.

In $n \times n$ matrix. the rational model approach of S-S in deriving common component and aggregation from the matrix of $\Psi(S^k_{ij})_{(1 \le i \le n,\ 1 \le j \le n)}$

In the equation $\Psi(S^k_{ij})\ X = \Phi^*(SC^k_i)\ X$. X denotes the eigenvector set; each set can be a pattern of common service. The implementation priority is based on the intensity of component reusability and frequency of fixed pattern. There is a possibility we may be able to use the above equation in deriving the patterns; however. how to best use a factor in balancing the left-hand side of the

1-4244-0885-7/07/$20.00 ©2007 IEEE.

equation which concerns rational model parsing and the right-hand side political model derivation is a concern which needs further development.

The scope can include business units, enterprises, and cross-enterprises within the same industry or cross-industry. The bigger the circle of semantic interoperability is, the more reusable the results will be in the long run but also more difficult in the areas of coordinating and reaching consensus. Moreover, an individual company will have less control over tailoring the reusable assets to its unique business needs and determining the desirable features. For these very reasons, some companies would use internal proprietary data models for their internal integration effort and use industry standard models for B2B. Enterprises should take a balanced view to consider the tradeoffs of adopting different patterns of semantic interoperability [3].

2. LITERATURE REVIEW

Based on the analysis of M. Selvage et al., once we have the priority of component implementation ready the roadmap is decided: at this point there are several different implementation patterns to achieve semantic interoperability. An obstacle to software reuse is the large number of major modifications that frequently have to be made as a consequence of dependencies within the reused software components. Component base implementation is the foundation in coping with this bottleneck [4]. Therefore, Composite Services is a use case driven parsing process; the purpose of the parsing iteration process is to discover the common areas and dependency between services. The determination of common component and service aggregation, as mentioned above, is through the parsing process by using use case driven model.

DSouza specified the standard and common component as follows: "To manage models of large scale, a coherent modeling architecture is required, with standard ways to describe shared concepts, rules, patterns, frameworks, mappings, and generators." For example, the development of different applications with object models mapped to relational tables requires a single object-to-relational mapping architectural style from a shared model as a refinement framework. If all the enterprise models maintain a separation of information, engineering, and computational views which are then integrated with them in particular ways, this common structure should itself be modeled in a shared architectural style model as a package pattern [5].

3. CONCEPT OF SERVICE AGGREGATION

Service aggregation is not merely component aggregation. There are two scenarios in producing service aggregation:

i) Business driven model: Before we decompose a business use case, we already know the service is common and critical. We do parallel implementation, one direction is to keep on the decomposition process to atomic level for producing common component for other service provisioning, the other direction is try to integrate each component in achieving the pre-defined coarse grained service, example is "open a new account".

ii) Component analysis model: This is a bottom-up case as before decomposition, we will never know the new aggregated service, and most of the cases are those business innovation or value added scenarios through component analysis, a typical example is: status query, balance query, security check, preference and service request are usually aggregated into a service interface called "customer service insight" coarse grained service. Figure 1. shows the ontology of these two aggregations. In component evaluation, for their significance in relation to common degree and level of aggregation, let S^* denote the finite set of services.

$S^* = \{ S_1, S_2, S_3, \ldots S_n \}, n \geq 2,$

Let S^{cc} denotes the finite set of services which qualified as a common component, we derive. $S^{cc} \subseteq S^*$.

In the evaluation of the significance factor of service component in K-B, B-S and S-C transformation, let

$P(X^k_j)$ be the jth member's weight for kth KPI

$\Pi(X^k_{ji})$ be the jth member's preference of component X_i for kth KPI

$\Phi(X^k_i)$ be the generic significance factor which is a concatenation between the weight of the role and preference of component X_i for kth KPI

$\Phi^*(S^{kb}_i)$ be the business significance factor in the B (business) dimension, which is a concatenation between the weight of role and preference of component S_i for kth K (KPI) and bth B (Business component)

$\Phi^*(C^{kb}_i)$ be the physical component significance factor in the I (information component) dimension, which is a concatenation between the weight of role and preference of component C_i for kth K (KPI) and bth B (Business component)

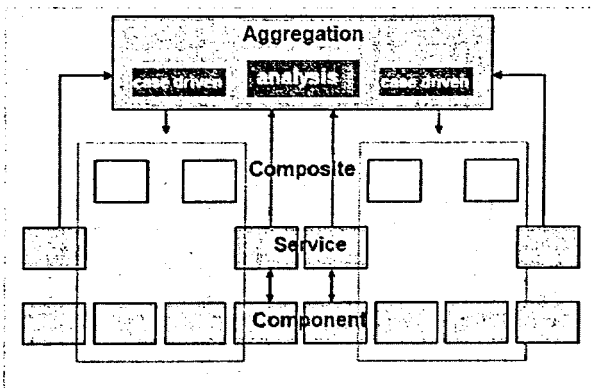


Figure 1. Ontology of aggregation

K-B transformation (size of Business component = n, size of decision group = m)

$\Phi(B^k_b) = \Sigma P(B^k_j)\ \Pi(B^k_{ji})$, let $\mu B^k = Max(B^k_b)$, denote the maximum value for B^k_b
$1 \le b \le n \quad 1 \le j \le m$

$\Phi^*(B^k_b) = \frac{\Phi(B^k_i)}{\mu B^k}$ be the relative value for B^k_b $(1 < i < n)$ when compared with μB^k

B-S transformation (size of Service component = n, size of decision group = m)

$\Phi(S^k_i) = \Sigma P(S^k_j)\ \Pi(S^k_{ji})$, let $\mu S^k = Max(S^k_i)$, denote the maximum value for S^k_i
$1 < i < n \qquad 1 < j < m$

$\Phi^*(S^{kb}_i) = \frac{\Phi(S^k_i)}{\mu S^k}$ $\Phi^*(B^k_b)$ be the relative value for S^k_i when compared with μS^k

S-C transformation (size of Component element = n, size of decision group = m)

$\Phi(C^k_i) = \Sigma P(C^k_j)\ \Pi(C^k_{ji})$, let $\mu C^k = Max(C^k_i)$, denote the maximum value for C^k_i
$1 < i < n \quad 1 < j < m$

$\Phi^*(C^k_i) = \frac{\Phi(C^k_i)}{\mu C^k}$ $\Phi^*(B^k_b)$ be the relative value for C^k_i when compared with μC^k

S-S transformation (CCSA parsing analysis)

Let $\Psi(S^k_{ij})$ $(1 < i < n,\ 1 < j < n)$ be the matrix from the result of n-dimension S-S parsing

Let $\gamma(S^k_{ij})$ denote the count of S^k_{ij} (dependency count from S^k_j to S^k_i)

When $i = j$, $\gamma(S^k_{ij})$ is the statistic of common degree of S^k_{ij}

When $i <> j$, $\gamma(S^k_{ij})$ is the statistic of dependency degree of S^k_j on S^k_i

Since $\Phi^*(S^k_i)$ $(1 < i < n)$ represents the significant factor of the service component using the political model, and the concatenation of K-B & B-S

$\Phi^*(C^k_i)$ $(1 < i < n)$ represents the significant factor of the service component using a political model, and the concatenation of K-B & S-C. This implies that the significance of service component S depends on both its associated business component and itself, as does component element C. The reason we use a parsing matrix to represent the significance factor is to try and derive dimensional vectors by means of different patterns. Both $\Phi^*(S^k_i)$ and $\Phi^*(C^k_i)$ statistics as an example of CCSA parsing analysis associated with atomic service components in provisioning upward service through the alignment of S-C, S-S and B-S transformation. Aggregation in the service layer is important as it eliminates boundary configuration, increases the integrity of common services in an error-free configuration environment and, furthermore, can be developed into coarse grain processes or processes adjacent to the business layer. Therefore the referential alignment of B-S strategy uses coarse grains to support business components directly; those fine grains at the bottom of the service layer can form a different pattern in supporting the coarse grains. This implies that the fine grains serving as the foundation of the services can be reused so that one service can contribute to provisioning many coarse grains through different patterns. The formation of a pattern can use common components as a base as they are common in provisioning generic services in each business component. Aggregation that is based on a pattern derived from parsing analysis as a usual case is crucial in common services: in the terminology our use of "common" indicates an encapsulated enterprise service that is seldom changed even though a light revision is acceptable.

1. At the very beginning of a use case, the first event triggers service Φ_1

$\varepsilon^1 : r_0 \times \Phi_1 \rightarrow r_1$

2. When the service event Φi ($i <> 1$) is read and triggers the service component, we express the state transformation as follows:

$\varepsilon^i : r_{i-1} \times \Phi_i \rightarrow r_i$ we further derive the transformation as follows:

$\varepsilon^{i-1} = (r_i) \times \Phi_{i+1} \rightarrow r_{i+1} = (r_{i-1} \times \Phi_i) \times \Phi_{i+1} \rightarrow r_{i+1}$

which is in equilibrium to $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$, and derives $\varepsilon^{nm} : r_n \times (\Phi_{n-1} \times \Phi_{n-2} \ldots\ldots \times \Phi_m) \rightarrow r_m$ $(n < m)$, we call this the "semantic event aggregator."

3. As the final step of a use case, the final event trigger service Φ_n

$\varepsilon^n : r_{n-1} \times \Phi_n \rightarrow r_n$ Φ_n is a terminator, an example of which is the settlement of a daily transaction.

In any component context a formal interface contract for the invocation syntax is important. The semantics issue (the meaning of parameters and so forth) has to be solved as well (domain modeling). This is crucial to any business-to-business (B2B) and dynamic invocation scenario. Such scenarios are cornerstones of the service component vision of being flexible and agile in response to the new business needs in a world of mergers and acquisitions, business transformation, globalization and other relevant factors. [6]

The string $\Phi(r_n, r_m) = \Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m$ is a path from r_n to r_m, we call the "service route" and $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$ a "composite service" between r_{i-1} and r_{i+1}, we use the symbol $\Phi_{i,\ i+1}$ to denote a composite service.

When frequency $F(\Phi_{i,\ i+1}) = F(\Phi_i \times \Phi_{i+1})$ is higher than threshold σ, we aggregate the concatenated alphabet (part of the string) $\Phi_i \times \Phi_{i+1}$ into $\Phi_{i\ ,\ i+1}$, this implies that the original state r_i between r_{i-1} and r_{i+1} will no longer exist for

this composite service.

4. CONCLUSION

Service aggregation is a concept that a composite service aggregated from at least two components that follow a certain sequence and perform a combination of coarse grained services. There are two type of aggregation: business driven model and component analytical model. The aggregation is validated when the analysis show a combination of services following a unique sequence frequently forming either an aggregated service or potential service-based process: the pre-built coarse grained service provides upward service provisioning and brings in its values in eliminating the integration interface and risk caused by boundary implementation. Once the aggregation services are built they can either be reused or aggregated to a coarser grain later if the aggregation is common and optimized in the parsing evaluation. These processes produce metadata interoperability for enterprise integration.

In K-B-S-C political model evaluation, a bottom-up approach using rational model parsing is applied to S-S transformation. This step is the last step in K-B-S-C transformation and the objective is to derive common components and service aggregation. The statistics of parsing results construct a matrix of common components and aggregated coarse-grained components which can be a candidate for process: this bottom-up approach attempts to create visibility of common services in either component units, interface or composite components with their priority and dependency. This provides the architects with a roadmap for building up their enterprise integration and strengthening the alignment and interoperability within the K-B-S-C transformation.

REFERENCES

[1] Curtis, K., Foster, P. W., Stentiford, F. (1999), "Metadata - The Key to Content Management Services", Proceedings of the Meta-Data '99. The third IEEE Meta-Data conference, April 6-7, 1999, Bethesa, Maryland, USA

[2] S. Dwarkadas K. Gharachorloo L. Kontothanassis D. Scales M. L. Scott R. Stets, (1998) ," Comparative Evaluation of Fine- and Coarse-Grain Software Distributed Shared Memory", Cambridge Research Laboratory

[3] M. Selvage, D. Wolfson, B. Zurek, E. Kahan, (2006), "Achieve semantic interoperability in a SOA", IBM Academy 2006

[4] L. Yu. (2005). "Categorization of Common Coupling in Kernel-Based Software," 43rd ACM Southeast Conference, 2005

[5] DSouza,(2001)," Model-Driven Architecture and Integration Opportunities and Challenges", Kinetium

[6] Traugott Koch, UKOLN, (2006), "Electronic thesis and dissertation services: semantic interoperability", E-Thesis Workshop. Amsterdam 2006-01-19/20

C3

C3: Common Component Evaluation and Micro Process Design

Publication: EISWT/ IEEE CS Proceeding (http://www.promoteresearch.org/2007/eiswt/index.html)
International Conference on Enterprise Information Systems and Web Technologies (EISWT-07)
July 9-12, 2007 Orlando, FL, 32819-8284, USA.

Publication Contact: Prof. John E. Edward
Editor in chief, EISWT
jeedward@gmail.com



"John Edward" <jeedward@gmail.com> Add to Address Book Add M

From: Yahoo! DomainKeys has confirmed that this message was sent by gmail.com more

To: rwu_academic@yahoo.com

Subject: EISWT-07

Date: Sat, 20 Jan 2007 18:00:50 -0500

Dear Colleague(s)

On behalf of the committee of the 2007 International Conference on Ent
Information Systems and Web Technologies (EISWT-07), I am pleased t
you that your paper: Common component evaluation and micro process
ID: eiswt107
has been accepted for presentation at the EISWT-07 conference to b
Orlando, FL, USA during July 9-12 2007. Congratulations! The paper
be included in the proceedings. The reviewers' comments are attach
email. The conference organizers should receive the revised and
formatted final paper, copyright form and the registration fee on or bef
2nd 2007. in order to include your paper in the conference proceedings.
Sincerely

John Edward
(on behalf of EISWT-07 committee)

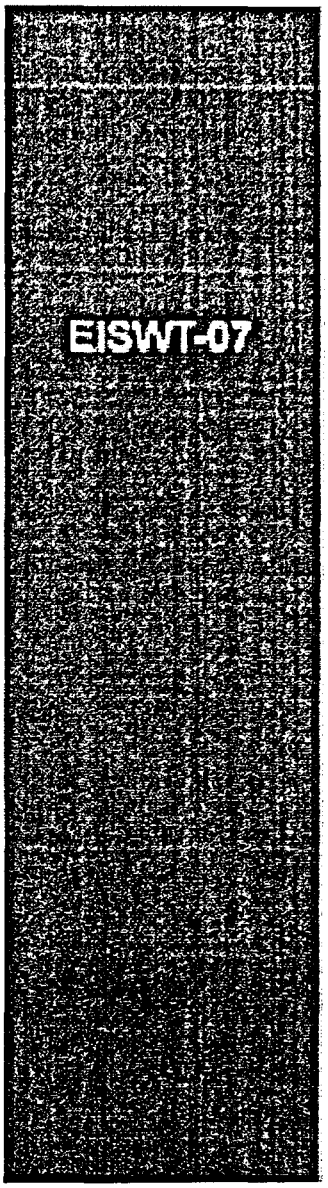
Acceptance/rejection will be based on the scores below. 1=worst, 5=bes

Relevance to conference (1-5): 5

Originality (1-5): 3-4

Contribution (1-5): 3

Technical quality (1-5): 3-4



PROCEEDINGS OF THE 2007 INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS AND WEB TECHNOLOGIES

Editors:
Bobby Granville, N.S.Kutti,
Michale Mesikoff and
Ngoc Thanh Nguyen

Associate Editors:
Scott Graham, Naphtali D. Rishe
and Booserm Wongsaroj

Orlando, Florida, USA
July 9-12 2007

EISWT 2007

ISBN 978-0-97741214-9

Contents

SESSION: Databases and XML
SESSION CO-CHAIRS: Scott Graham, Naphtali D. Rishe and Booseria Wongsaroj

Common Component Evaluation and Micro Process Design 47
Raymond Wu

Common component evaluation and micro process design

Raymond Wu
IBM Software Group
raymond@de.ibm.com

Abstract

Service component implementation requires micro process design and interoperability strategy. Our enterprise integration in a component-based environment follows the pre-built framework and life cycle of decision, analysis, design and virtualization. So once we finished the component identification by using political model decision-making and rational model analysis, we shifted our focus to the design stage which covers both inter-component micro process and interoperability across components. We applied SCA architecture framework, dimensional patterns and aggregation concept in achieving our goal. We believe that although both micro and macro processes can be built independently, a macro design view is required in handling micro interoperability; our literature review indicates that application of much of the existing research may be infeasible due to lack of a macro view in micro design, or weakness in macro design unsupported by a micro process view. For this reason, our integration skeleton strategy takes business semantics and SCA to micro process design; we apply parsing techniques in aggregation to construct a loosely coupled micro process in provisioning the micro service process. This is the foundation of service virtualization.

Introduction

The micro process of enterprise integration differentiates itself from the macro process in that the macro process is working on the configuration of components at an architectural level while the micro process deals with components both internally and in their interference in component fabrication, with how they are integrated in micro mechanisms and with how they work together in provisioning the coherence of a business goal. This includes the integration of common components and service aggregation in micro processes.

Both micro and macro processes are important to enterprise integration. Our integration strategy and framework presented in a separate research paper indicated that the transformation from business context to service component cannot be covered solely by the deconstruction process; the business semantic injection in micro process is crucial due to each component being responsible for a particular business rule. In a component-based implementation there is no duplication or mix-up allowed in component design; if the dependency between particular components is high and they always follow the same sequence in their referential integrity, we call this a good candidate for service aggregation [1]. Those components with a high frequency of utilization are termed common components. Both aggregated and common components need to be handled carefully.

A component-based model builds the foundation for both business and service components, storing the intelligence of mapping and interconnectivity information in a repository in metadata form and strengthening the foundation of vertical enterprise integration. There are three areas of micro process needing to be implemented:

1. Micro mechanism of service component fabrication and interference
2. Micro process in component transformation within the service layer
3. Interoperability in enterprise integration and service virtualization

Literature review

According to P. Kruchten, a component is a non-trivial, nearly independent and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture. A component conforms to and provides the physical realization of a set of interfaces. Metadata enhancement in the component context of services, after initial creation, is an important activity. [2]

i) A component is non-trivial; it is functionally and conceptually larger than a single class or a single line of code. Typically, a component encompasses the structure and behavior of a collaboration of classes.

ii) A component is nearly independent of other components. It rarely stands alone. A given component collaborates with other components and in so doing assumes a specific architectural context. This architectural context is driven in large part by the implementation we choose.

iii) A component is a replaceable part of a system. A component is substitutable for any other component that realizes the same interfaces. This aspect helps during development, in that parts of a system can be stubbed, sketched and then replaced by mature, robust implementations. It also supports the evolution of a system even after it has been deployed by making it possible to upgrade and evolve parts of the system independently.

iv) A component fulfills a clear function. It is logically and physically cohesive and thus denotes a meaningful structural and/or behavioral chunk of a larger system. It is by no means an arbitrary grouping.

v) A component exists in the context of a well-defined architecture. A component represents a fundamental building block upon which systems can be designed and composed.

This definition is recursive: A system at one level of abstraction may simply be a component at a higher level of abstraction, yet components never stand alone. Every component presupposes an architectural and technology context in which it is intended to be used. Finally, a component conforms to a set of interfaces. A component conforms to a given interface when it satisfies the contract specified by that interface and may be substituted in any context to which that interface applies [3].

Our interest in the micro process starts with the mechanism for how the component is designed; this involves taking the business semantic of the macro level with coherence of enterprise interoperability between each layer [4] [5].

Before the design of the service component, we take direction from the business component by using a top-down approach to derive the component design while at the same time streamlining the alignment between service components and physical components by using both top-down and bottom-up approaches. Most of the bottom-up approaches are a componentization process working from existing information components, business objects, codes, system metadata or database access protocols. Service aggregation means a composite service aggregated from at least two components that follow a certain sequence and perform a combination of coarse grained services. Service aggregation is validated when the parsing statistics show a combination of services following a unique sequence frequently forming either an aggregated service or potential service-based process; the pre-built coarse grained service provides upward service provisioning and brings in its values in eliminating the integration interface and risk caused by boundary implementation [6].

Once we construct the alignment between business, service and information and have the service component design and interconnectivity completed, we actively work on service layer transformation. The reason this step is crucial is that the goal of vertical enterprise integration is to create interoperability between layers from both micro and macro perspectives, with both business and information layers working as a foundation to support the formation of the service layer which provides the actual services. Once the service component has completed its design with alignment to business and information, the focus can be shifted to micro interference within the macro view. Once the alignment is missing or invisible between layers, the common services and service virtualization become impossible as these require a foundation in both macro interoperability and micro mechanism. This results in a silo system with serious limitation of services and weakness in enterprise integration.

Service component architecture and micro process design

The component element is the physical element that mediates service and information. The service component consists of interface protocol, business rules, semantic intelligence and referential information. It is important to note that (a) components are encapsulated for reuse; (b) components may be wired together as composite services; and (c) business objects are the data flowing on wires between components. The design of service components follows our previous strategy by using business semantic injection to take the context into the semantic and micro processes; by this means each component has its contract-level responsibility in provisioning a subset of service. The micro process strategy we are introducing here is called SCA (service component architecture) and provides a framework for service component micro process and interconnectivity design.

SCA deals with component design and interconnectivity in micro mechanism and consists of three main areas. The service component is produced according to the following three criteria, and the SCA framework is shown in Figure 1.

1. Interface

The service interface specifies SP (service provider) implementation and is invoked by the SC (service coordinator). The SC has the capability to invoke SPs through the service bus without needing to know what the SPs are and how the interface was configured.

Luca et al. demonstrated their surveying results from the theory behind formalizing rich interfaces for software and hardware components. Rich interfaces specify the protocol aspects of component interaction. Their formalization, called interface automata, permits the compiler to check the compatibility of component interaction protocols. Interface automata support incremental design and independent implementation. Incremental design means that the compatibility checking of interfaces can proceed for partial system descriptions without knowledge of the interfaces of all components. Independent implementation means that compatible interfaces can be refined separately while still maintaining compatibility. We used both of these methodologies in component interface design [7].

2. Implementation

The service provider implementations in an SCA are called service components and include:

- Java objects

A Java object is an implementation of a component in Java which has either a WSDL (web service definition language) or Java interface type. A Java object normally works in a mediation module.

- Business processes

A business process is a combination service made up of service components with a specific order to achieve a business goal; a business process should not be equal to a use case, rather it can be a result of deconstruction and composed of use cases.

- Variables

Variables store the data used by a business process. If we conceive of the messages as representing the writing, then the variable represents the paper.

- Microflow or short-running processes

A microflow is a process that is contained within a single transaction. This is ideal for situations where the user is expecting an immediate response.

- Long-running processes

A long-running process executes over an extended period of time and is much more flexible and resilient than a microflow. Interruptible and asynchronous business processes are examples of long-running processes.

- Human tasks

A human task is a unit of work done by a human which involves interaction with other services; it thus becomes a task within a larger business goal.

- Business state machines

A state machine is an event-driven business transaction in which external operations trigger changes that guide the transaction from one discrete mode to another. Each mode is an individual state, and this mode determines what activities and operations can occur.

- States

A state is one of several discrete individual stages that are organized in sequence to compose a business transaction.

- Composite states

A composite state is an aggregate of one or more states. It is used to deconstruct a complex state machine map into an easily comprehended hierarchy of state machines.

- Business rules

A business rule is anything that captures and implements business policies and practices. A rule can enforce business policy, make a decision, or infer new data from existing data; the component design is shown in Figure 2.

The parameters transmitted in and out from a service are modeled as service data objects implemented by Java class. SDO (service data object) complements SCA by providing a common way to access many different kinds of data. The specification reduces the skill level and time required to access and manipulate business data. Today, a great many APIs are used to manipulate data. These APIs tend to tightly couple the source and target of the data; this makes their usage error-prone and subject to breaking as business requirements evolve. However, SDO makes it easier to use, and thus realize the value of, these APIs without having to code directly to them [8].

Let $\Phi^*(SC^k_i)$ denote $(\Phi^*(S^k_i) + \Phi^*(C^k_i))_{(1 < i < n)}$, and represent the vertical intensity of service component i (under strategy k) in both upward business and downward information dimensions; $\Phi^*(SC^k_i)$ is actually a matrix produced by the political model. Meanwhile, we introduce $\Psi(S^k_{ij})$ to denote the matrix of counts from our use case parser.

From an interoperability perspective, as the objective is to build the enterprise's aggregated services for interoperability, assume the value we derived from the political model $\Phi^*(SC^k_i)$ satisfies the eigenvalues approach and works as an eigenvalue of the rational model. In n × n matrix, the rational model applys approach of S-S in deriving common component and aggregation from the matrix of $\Psi(S^k_{ij})_{(1 < i < n,\ 1 < j < n)}$

In the equation $\Psi(S^k_{ij})\ X = \Phi^*(SC^k_i)\ X$, X denotes the eigenvector set; each set can be a pattern of common service. The implementation priority is based on the intensity of component reusability and frequency of fixed pattern. There is a possibility we may be able to use the above equation in deriving the patterns; however, how to best use a factor in balancing the left-hand side of the equation which concerns rational model parsing and the right-hand-side political model derivation is a concern which needs further development [9].

Mediation patterns manipulate messages in-flight on the bus by requests or events. According to IBM service mediation, there are several basic patterns for mediations; the following more complex patterns can be built by combining the simple patterns.

• Protocol switch: Enables service requesters to dispatch their messages using a variety of interaction protocols or APIs such as SOAP/HTTP, JMS and the MQ Integrator (MQI).

• Transform: Translates the message payload (content) from the requester's schema to the provider's schema. May include enveloping, de-enveloping or encryption.

• Enrich: Augments the message payload by adding information from external data sources such as customization parameters defined by the mediation, or from database queries.

• Route: Changes the route of a message, selecting among service providers that support the requester's intent. Selection criteria can include message content and context as well as a target's capabilities.

• Distribute: Distributes the message to a set of interested parties and is usually driven by subscriber interest profiles.

• Monitor: Observes messages as they pass through the mediation unchanged. Can be used to monitor service levels, assist in problem determination or meter usage for subsequent billing to users.

• Correlate: Derives complex events from message or event streams. Includes rules for pattern identification and rules that react to pattern discovery, for example by generating a complex event derived from the content of the triggering event stream.

Once the vertical skeleton and service extension are built, the interoperability reflected from the business layer has the capability of integrating cross-organization and cross-geographical constraints, as fully transparent services become possible if each system or organization follows an industry standard by using metadata strategy and an industry framework. An example of this would be a case in which several key industry vendors have built up an industry framework for service providers in that particular industry to follow, starting from business domain analysis and working through to component deployment. Each service provider would thus benefit from the industry's common services and the common interface, particularly those involving merger and acquisition cases (B2B, B2C). Any business activity becomes easier when the various parties involved have a common understanding and a consensus on business terms, service definitions, interfaces and component specification, particularly if these become transparent and common practices. The important concept here is that once the metadata of business, service and information constructs an industry standard, the organization's vertical skeleton has the opportunity to integrate across multiple organizations for an easier global practice; this is a long-term industry objective which we call service virtualization in a fully transparent environment [10].

The interoperability skeleton shown in Figure 3 is an example of how the micro interoperability in the service layer, aligned with the component layer in micro mechanism, provisions the macro service in business. The foundation of this skeleton is achieving interconnectivity between service components by using a common component and an aggregated component. We further develop the service virtualization on top of this interoperability

skeleton which we call enterprise vertical integration. in extending the integration across industry and geographical areas. This is the virtualization service in a horizontal direction: once the industries of different regions build up their semantic interoperability in three metadata repositories (business. service and information). they can synchronize their business, operation and strategy even if they have their local implementation extended from the interoperability skeleton. This is what we call service virtualization; we predict that virtualization of services will become a priority in industry as semantic interoperability is in high demand to solve the issues in both vertical interoperability and horizontal interconnectivity.

Conclusion

Enterprise interoperability is built on the foundation of a validation process of decision. design and provisioning stages. In this paper we focused our research on micro process design; this encompasses component-based design and interconnectivity, transformation of service aggregation through rule-based design. service provisioning in a particular dimension and service virtualization.

In component design we introduced the SCA (service component architecture) to further clarify the micro mechanisms within components as well as component interconnectivity. We classified component design into interface, implementation and metadata areas to guide the design from the internal to the external.

In transformation of service aggregation we explored the possibility of applying automata theory in service aggregation. We found that the concept of automata theory can be applied to service aggregation in building up rule-based aggregated services if we handle the service execution by using the same concept of symbolic reading. Handling the parser in use case parsing with the same machine reading concept, the state concept can also be applied to the state of service layer with metadata alignment to the business and information layers.

In enterprise service provisioning and virtualization, we constructed the skeleton of enterprise service with leverage to each layer and the interoperability of the layers. Service virtualization is a process of leveraging enterprise integration strategy within a specific geographical region or across regions. This can be an industry-standard scenario such as a credit card transaction in which the service, geography and systems are fully transparent to both cardholders and merchants. A full virtualization environment needs to be built on top of the foundation of component design and service transformation process in order to apply the vertical enterprise integration approach to this final stage which is the goal of our research work. The standardization of metadata technology and common services and components are all requirements in this stage.

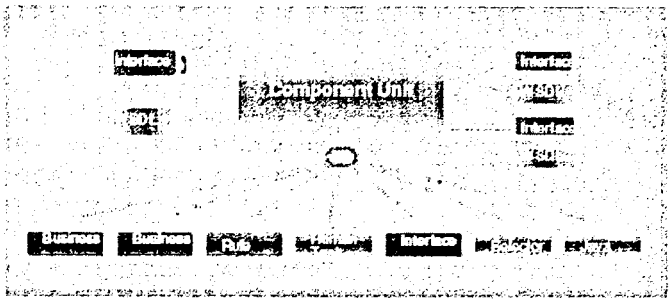


Figure 1. Micro process component design

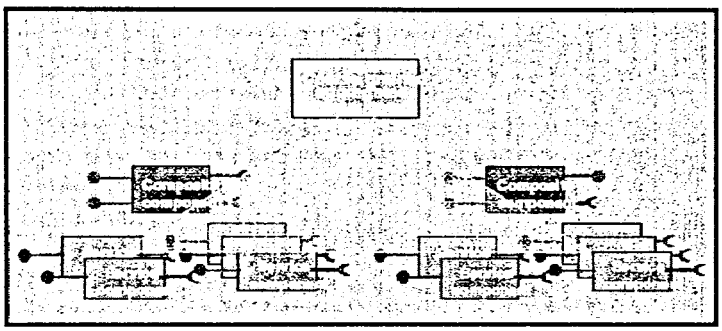


Figure 2. Micro process component assembly

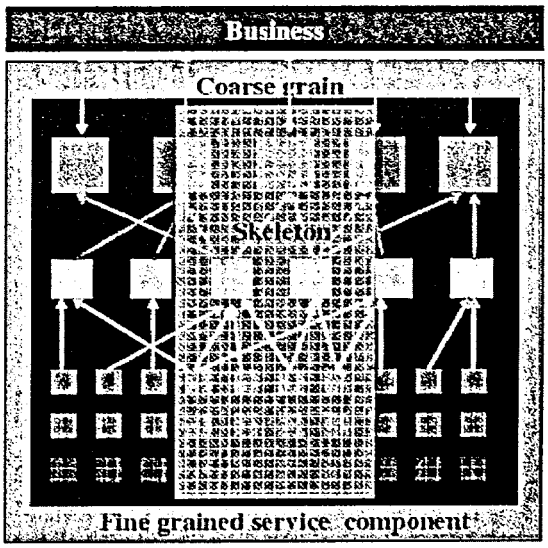


Figure 3. Micro process service component alignment

References

1. Remko Scha, Rens Bod[3] and Khalil Sima'an, (1999), "A Memory-Based Model of Syntactic Analysis: Data-Oriented Parsing" Special Issue on Memory-Based. Language Processing) 11, pp. 409-440.
2. Traugott Koch, UKOLN, (2006), "Electronic thesis and dissertation services: semantic interoperability", E-Thesis Workshop, Amsterdam 2006-01-19/20
3. Philippe Kruchten, (1998), "Modeling Component Systems with the Unified Modeling Language", 1998 International Workshop on Component-Based Software Engineering
4. O. Valkeapää and E. Hyvönen, (2006), "Semantic Annotation with Browser-based Annotation Tool", 1st Asian Semantic Web Conference
5. J. Hunter , (2001), "MetaNet - A Metadata Term Thesaurus to Enable Semantic Interoperability Between Metadata Domains", Journal of Digital Information, Volume 1 Issue 8
6. S. Dwarkadas K. Gharachorloo L. Kontothanassis D. Scales M. L.Scott R. Stets, (1998) ," Comparative Evaluation of Fine- and Coarse-Grain Software Distributed Shared Memory", Cambridge Research Laboratory
7. L. de Alfaro and T. A. Henzinger, "Interface theories for component-based design," in *Proceedings of the First International Workshop on Embedded Software* (EMSOFT), Lecture Notes in Computer Science 2211,Springer-Verlag, 2001, pp. 148-165.
8. DSouza,(2001)," Model-Driven Architecture and Integration Opportunities and Challenges", Kinetium
9. L. Yu, (2005), "Categorization of Common Coupling in Kernel-Based Software," 43rd ACM Southeast Conference, 2005
10. H. Lee and J Lee, (2006), Analyzing Business Domain: A Methodology and Repository System, Korea Advanced Inst. of S&T

C4: Componentization and Semantic Mediation

Publication: IECON2007 Journal
http://iecon07.ccu.edu.tw/

Publication Contact: Ren C. Luo, IECON'07 General Chair
IECON2007@ia.ee.ccu.edu.tw

IEEE IECON 2007

From: "IECON 2007" <IECON2007@ia.ee.ccu.edu.tw> Add to Address Book Add Mobile Alert

To: "Raymond Wu" <rwu_academic@yahoo.com>

Subject: IECON07

Date: Fri, 22 Jun 2007 10:35:13 +0800

TD-018309
Componentization and semantic mediation

We are pleased to inform you that your paper referenced above has been accepted for presentation at the 33rd Annual Conference of the IEEE Industrial Electronics Society (IECON'07), to be held in Taipei, Taiwan, R.O.C., Nov. 5-8, 2007. This letter is a formal and personal invitation for you and your co-authors to participate, and it is also an official document of the conference. All authors of the paper can use this letter to obtain from their institutions any approvals that may be required to attend.

As the corresponding author, you are the only person receiving this information. You are kindly requested to notify all co-authors about this correspondence. We look forward to meeting you in this November, for a fruitful Taipei conference.

Kindest regards,

Ren C. Luo, IECON ' 07 General Chair
Han-Pang Huang, IECON ' 07 Program Chair
Terry Martin, Carlo Cecati, and Yoichi Hori, IECON ' 07 Program Co-Chairs

IECON07 Manuscript Submission System

Comments from Reviewers Regarding Your Manuscript

TD-018309

Feedback from the Reviewers

- A. Contribution and Clarity:
  importance of scope ---------- [4 - Significant]
  original result ---------- [4 - Significant]
  application oriented ---------- [4 - Significant]
  clear and concise ---------- [3 - Minor]

  B. Recommendations to TPC ---------- [4 - Should be included]

  C. Suggested form of presentation -- [O - Oral presentation]

- A. Contribution and Clarity:
  importance of scope ---------- [4 - Significant]
  original result ---------- [4 - Significant]
  application oriented ---------- [4 - Significant]
  clear and concise ---------- [4 - Significant]

  B. Recommendations to TPC ---------- [4 - Should be included]

  C. Suggested form of presentation -- [O - Oral presentation]

- A. Contribution and Clarity:
  importance of scope ---------- [4 - Significant]
  original result ---------- [4 - Significant]
  application oriented ---------- [5 - Exceptional]
  clear and concise ---------- [5 - Exceptional]

  B. Recommendations to TPC ---------- [4 - Should be included]

  C. Suggested form of presentation -- [O - Oral presentation]

Comments:
Good Paper. The paper mainly looks like a review paper. However given the importance of this topic and the theme of this conference, it can be accepted. Please ensure that should this paper be accepted, the authors comply with the IEEE paper formatting requriments.

Componentization and Semantic Mediation

Raymond Wu
IBM Software Group
raymond@de.ibm.com

Abstract

Reuse and agility are all basic requirements in a rapid development environment. Among all disciplines, componentization is crucial and also the foundation to maintain a consistent responsiveness between business and IT. Based on this objective, this paper proposes that first the terminology of semantic mediation needs to be clarified. This should not be initiated from the macro architecture layer; instead we propose that both mediation and interoperability should impact components first and push semantics strategy directly into the micro process to be synthesized into the service layer. To achieve this goal, a precise framework for its life cycle includes, decision, analysis and design are key steps in building the skeleton of enterprise integration. In this research paper we deal with the component identification, we further explore the brief of micro mechanism of component interference and service provisioning in achieving service virtualization.

Introduction

A component represents a fundamental building block upon which systems can be designed and composed. This definition is recursive: a system at one level of abstraction may simply be a component at a higher level of abstraction, yet components never stand alone. Every component presupposes an architectural and technology context in which it is intended to be used. Finally, a component conforms to a set of interfaces. A component that conforms to a given interface means that it satisfies the contract specified by that interface and may be substituted in any context to which that interface applies. [1] Components cannot work as silo, they require mediators, in either virtual or physical form, to transmit the semantics and maintain the interoperability coherence within enterprise. The behavior of those components work as semantic mediators, we call semantic mediation, thus the ontology concerns enterprise interoperability requires both componentization and semantic mediation to synthesize component in both micro and macro view.

Our integration approaches are based on the metadata component which is built on top of the foundation of semantic interoperability. This means our method is not based on peer-to-peer data or system or application integration, rather our interest is in following the nature of business. This interest includes how the business is composed and what strategy synthesizes the business and technology aspects into two directions made up of architecture strategy and the project life cycle. We observe and use a working technology to construct the strategic "framework" and "mechanism" as the foundation of service interoperability [2]. We believe the nature of business implementation and human behavior need to be taken into account as these can never be replaced by merely using technical assets. Our final goal is service virtualization; this concept indicates that in the virtualization stage enterprise integration reaches its highest maturity and seamless integration across enterprise and industry. We believe

this objective is one of the key tasks in this century for most organizations; industry requires a common objective and roadmap to achieve this goal, thus researchers working from the subject of point solution can complement each other in creating a homogeneous environment.

A component conforms to and provides the physical realization of a set of interfaces. First, a component is non-trivial; it is functionally and conceptually larger than a single class or a single line of code. Typically, a component encompasses the structure and behavior of a collaboration of classes. Second, a component is nearly independent of other components. It rarely stands alone. A given component collaborates with other components and in so doing assumes a specific architectural context. This architectural context is driven in large part by the implementation we chose. Third, a component is a replaceable part of a system. A component is substitutable for any other component that realizes the same interfaces. This aspect helps during development in which parts of a system can be stubbed, sketched and then replaced by mature, robust implementations. It also supports the evolution of a system even after it has been deployed by making it possible to upgrade and evolve parts of the system independently. Fourth, a component fulfills a clear function. It is logically and physically cohesive and thus denotes a meaningful structural and/or behavioral chunk of a larger system. It is by no means only an arbitrary grouping. Fifth, a component exists in the context of a well-defined architecture.

Literature review

According to P. Kruchten, a component is a non-trivial, nearly independent and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture. Desmond DSouza defines vertical integration in his "Model-Driven Architecture and Integration Opportunities and Challenges," stating that vertical integration comes from different levels of abstraction of the same system, from physical data models, logical data models, networks, applications specifications, component assemblies, business process models, and business goals and strategies [2].

Common interface is also a means to achieve common components. S. Melnik proposes a universal interface which avoids common models and languages; this is the Generic Interoperability Framework which was developed to facilitate the integration of heterogeneous information systems. Within this framework individual communication protocols, data manipulation languages and data are represented in a generic manner while preserving their ontological variety; this process indicates how a mediation infrastructure can be built using canonical wrappers, operational requirements of integration and layered architecture for component design [3]. D. Oberle, S. Staab and A. Eberhart, in their Kaon server design, show several potential sources of semantic metadata. These sources provide input for the framework regarding the semantic management of software components. This includes deployment descriptors, Web and application server configuration files, source code information or annotations and database metadata. The component metadata collector parses this information and automatically integrates it into the ontology as semantic metadata (instances). The inference engine is embedded in the application server and applies the ontology and semantic metadata for reasoning and querying. The application server's core functionality and the administration console both use the reasoning capability. The latter allows the developer to browse and query the ontology at runtime so that configurations may be

queried or checked to avoid inconsistent system configurations during development and at runtime. [4]

We define our semantic mediation as the means of semantics transmission through multi-dimensional approaches. First dimension is component validation between human and machine through decomposition and aggregation processes. Second dimension is semantic transmission through metadata design and synthesis of micro and macro processes. The third dimension is semantic implementation under enterprise vision, so once the three levels of semantic mediation is implemented, it gives opportunity for atomic components be designed in a synthesis way which consists of micro and macro processes. Meanwhile, as the visibility of vertical alignment is clear between architecture layers, the semantic centric metadata forms a clear picture of the enterprise skeleton stretched across the interoperability between business, service and information. Through a semantic mediation approach, we can either reuse the metadata mapping and configuration from that successful experience or construct a new map for the new requirements by using the same methodology. Consequently, once the service components and aggregation services are built they can either be reused or aggregated to a coarser grain if the aggregation is common and optimized in the parsing evaluation.

Component identification

The design of service components follows our previous strategy by using business semantic injection to take the context into the semantic and micro processes; by this means each component has its contract level responsibility in provisioning a subset of service. This interface helps the coordination between components and metadata to build the foundation of interoperability within the service layer and between layers. Figure 1. shows the component identification initiated from business component. In the following paragraph we take the results from component design into the service level while applying automata theory to build up the rule-based service aggregation at the enterprise level. Figure 2. shows the decomposition and aggregation.

Beneath the service component, the component element is the physical element that mediates service and information. The service component consists of interface protocol, business rules, semantic intelligence and referential information.

It is important to note that (a) components are encapsulated for reuse; (b) components may be wired together as composite services; and (c) business objects are the data flowing on wires between components.

Semantic mediation and service transformation

The transformation therefore requires an event to incur the service with the intelligence to either predict the next service or to verify that the called service is recognized. We use the state concept discussed above to mediate the business, service event and information in the "time" dimension. The state of the service layer and the business-service-information coherence is changed once we trigger an event and the service is successfully executed. The alphabet read in automata is leveraged to service component trigger and execution while the state change is leveraged to transformation within the service layer.

In any component context a formal interface contract for the invocation syntax is important. The semantics issue (the meaning

of parameters and so forth) has to be solved as well (domain modeling). This is crucial to any business-to-business (B2B) and dynamic invocation scenario. Such scenarios are cornerstones of the service component vision of being flexible and agile in response to the new business needs in a world of mergers and acquisitions, business transformation, globalization and other relevant factors. [5]

A tool called JFLAP guides the user through several proofs of theorems that focus on transformations of languages from one form to another in which the forms are automata and grammars. JFLAP aids in automata visibility; users are given a visual representation they can interact with in addition to the formal textual representation. This alternative hands-on representation provides a means for an interactive automata theory. [6]. Virtual coarse grained service eliminates the boundary and leverages many fine-grained components which can be easily configured by adding or modifying the component dynamically to achieve the broader coverage of main route services. For this reason main route is one of crucial services in enterprise integration. One example is in the Telecommunication industry that incorporates an open service account, an open billing account and provisioning options for a new customer.

In our literature review regarding this research field, we saw that some crucial points were missing. We consider this crucial because within enterprise, the skeleton of dimensional service with linkage between business, service, component and information is mandatory. By means of these dimensional services, enterprise and industry have the opportunity to provision their strategic dimensions with further impact on patterns and common components. The assumption is that industry requires a certain level of maturity in common practices with regard to service dimensions and the derivation process in achieving realization. A typical example is the eTOM level 3 in the telecom industry. Our priority in service transformation is the knowledge of how to create the visibility of the alignment within these dimensions and how the components, either in conceptual, logical or physical forms, mediate each other with linkage to the skeleton of service architecture. We use the following two terminologies to represent semantic notation:

1. Semantic mediation: The virtual semantic service in mediating components or contents in provisioning combinational services. The macro view of semantic mediation is related to service dimension which means many fine grain services aggregated into a coarse grain in particular business dimension to perform a particular service, in the other word, a group of service components and alignment in provisioning a particular business objective; examples include customer insight and pre-paid services. A service dimension is different from a use case, as use case is in daily operational form which is a virtual process directly decomposed from the business block, while the service dimension is a strategy for achieving a strategic business feature.

2. Service pattern: The pattern used in provisioning a particular service dimension. Pattern is a symbolic combination of components used by following a particular rule or principle. We wish to demonstrate how the service dimension is represented by a mathematical model. The strategy and realization of architectural interoperability by using semantic mediation strategy in provisioning the enterprise level service. This constitutes a main route for "new customer service" coarse grained service, though it is aligned with fine-grained components to support customer preference,

service criteria and billing options which are branches within this main route.

The importance of determining the common component and service aggregation is in building up an enterprise skeleton of vertical interoperability between business, service and information. This task is crucial, as in the business domain analysis stage, the impact area and scope to business components are minimized in decision making.

Additionally, the business component is deconstructed into service components and physical elements for top-down alignment while the common component is not simply a symbolic service but instead requires a bottom-up approach to work on common areas; this serves to fine tune the common and interfaces from the perspective of enterprise integrity and performance.

The service aggregation is also important as the top-down deconstruction followed by bottom-up composition forms the skeleton of vertical interoperability and potential component-based implementation of the service process.

Conclusions

Our semantics and mediation approaches differentiate itself from conventional concept due to following reasons: The semantics concept we introduced consists of three level derivation: i) technical level in atomic element form, ii) syntax level about elements' interconnectivity and iii) context level which is environment dependent. None of these three levels can be omitted as the alignment within these three levels is the key to take human decision and vision into technical realization.

To comply with service mediation principles, there are two significant issues which need immediate action: first, enterprise misalignment which, as we pointed out in previous research, indicates that an enterprise loses governance in semantic interoperability in the maintenance of coherence between the business, service and technical components and second, semantic mediation, which has been misinterpreted by several activities and needs a new direction in order to comply semantic mediation strategy to the following requirements:

1. Define the scope of enterprise interoperability
2. Component analysis and mediation
3. Using repository to hold the metadata, patterns and components for future reuse
4. Unless there is a certain level of intelligence built on top of empirical knowledge to aid in service pattern automation

The barrier from theory to realization is high because the gaps, such as language difference, geographical distance, and enterprise terminology between business and technology are all bottlenecked when attempting to move the micro mediation outward to macro mediation. For these reasons tools, software and framework are all key factors in our roadmap.

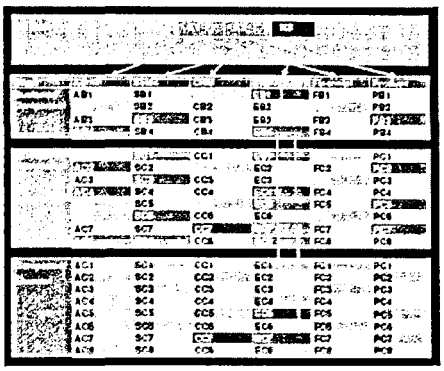


Figure 1. Component identification

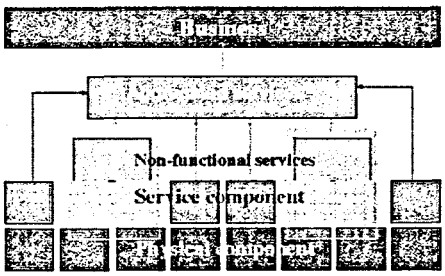


Figure 2. Semantic injection and aggregation

Reference

1. Philippe Kruchten, (1998), "Modeling Component Systems with the Unified Modeling Language", 1998 International Workshop on Component-Based Software Engineering
2. DSouza,(2001),"Model-Driven Architecture and Integration Opportunities and Challenges", Kinetium 2001
3. Sergey Melnik, (1999), "Generic Interoperability Framework", Distributed and Parallel Databases Journal 1999
4. D. Oberle, S. Staab and A. Eberhart, (2006), "Semantic Management of Distributed Web Applications", IEEE Distributed Systems Online, vol. 7, no. 4, 2006, art. no. 0605-o5001
5. O. Zimmermann, P. Krogdahl, C. Gee, (2004), "Elements of Service-Oriented Analysis and Design", IBM Academy
6. E. Gramond and S. H. Rodger, (1999), "Using JFLAP to Interact with Theorems in Automata Theory", Thirtieth SIGCSE Technical Symposium on Computer Science Education, ACM Press 1999 page 336-340.

C5: Semantic Interoperability and Transformation in Enterprise Integration

Publication: ACM NZ
ISSN 1176-9998 (http://www.acm.org/chapters/acm_nz/bulletin)

Publication Contact: Taiyu Lin, Inf. Systems
Department, Massey University,
New Zealand t.lin@massey.ac.nz



Association for Computing Machinery New Zealand Bulletin

Date: Sun, 01 Jul 2007 19:20:48 +1200

From: "Taiyu Lin" <taiyu.lin@gmail.com> Add to Address Book Add Mobile
Yahoo! DomainKeys has confirmed that this message was sent by gmail.com

To: "Raymond Wu" <rwu_academic@yahoo.com>

Subject: Re[2]: acm_nz: submissions

Dear Raymond,

I am pleased to announce that your paper titled " Semantic interoperability and transformation in enterprise integration" has bee accepted by ACM_NZ Bulletin.

As the previous paper, please be aware that the acceptance is based on the condition that you carefully revise your paper according to reviewers' comments especially about the readability.

You will need to document how you treat reviewers' comments in your revised manuscript. I hope that we can receive your revised manuscript soon.

p.s. Reivewerl's comments are in the attached document.

Best regards.
Taiyu

Semantic interoperability and transformation in enterprise integration

Raymond Wu
IBM Software Group
raymond@de.ibm.com

Abstract: *The main purpose of enterprise interoperability is to try to cope with changeable business and multiplicity of technologies by means of component identification, design, interconnectivity and service transformation. A rough roadmap has been initiated by consolidating all heterogeneous environments and system silos into a component-based foundation. In applying particular methodology, framework and design to implement common practices, the transformation requires different technologies and vision to achieve service virtualization – which is a new area in the future service industry – to form a common main route. The service grid concept will provision a transparent and highly integrated framework to provide a global service infrastructure which is not limited by geography or by the system in use.*

Keywords: Semantic interoperability, Enterprise integration, Service transformation, Virtualization

Introduction

Service transformation follows a common framework with common services and components for service consumers to share common business components, service components, information schema and service bus in a transparent environment. Common components and enterprise interoperability are pre-requirements in this transformation. The reasons for this are:

i) In current technology, there is no way to achieve "common practice" unless the componentization foundation is solid. This foundation offers flexibility in implementing service components, as each atomic element has its clear boundary with the next; once they are identified and specified, component micro design is feasible and component interconnectivity can be easily defined.

ii) We define micro interoperability as "component internal design from a macro architecture view, and macro architecture design from micro interconnectivity view"; this means the metadata not only carries information, it plays a semantic mediation role in both micro and macro design.

iii) In the service transformation, we need to prevent a "big bang" as most of the industry is not ready to take the risk of a fundamental change. We need to create intermediate steps for the enterprise to transform their business and technology. Most of the roadmap is a bottom-up approach in consolidating infrastructure, application and business components, as shown in Figure 1.

As aggregated services and common components are critical in forming the enterprise skeleton, the design and aggregation process must be cautious from an interconnectivity perspective. This implies that high integrity, optimization, grain size, dimension in service provisioning, visibility, traceability, performance and other factors are areas which must be brought into consideration. An example is the credit card business, in which aggregated services have been built between the card holder and card issuer; in this case the same aggregation is also being formed between the merchant and the bank. The mediation service of an inter-banking fraud detector component has been designed by reading fraud profiles in real-time transaction environments to mediate the

individual services into a business main route; this highly enhances the service provisioning in a particular dimension.

Literature review

It is becoming generally accepted in the information community that interoperability is one of the most important principles in metadata implementation. Other basic metadata principles include simplicity, modularity, reusability and extensibility (Duval et al., 2002; Zeng et al., 2003). These principles inform metadata database design as well as other system-dependent developments. From the very beginning of a metadata project, the principles that enable user-centered and interoperable services should be foremost in design and implementation. Semantic interoperability goes beyond attempts to homogenize disparities through standards (Harvey, 1999). A central question about semantic interoperability is how people and social groups with different perspectives identify and possibly resolve their semantic differences (Harvey, 1999).

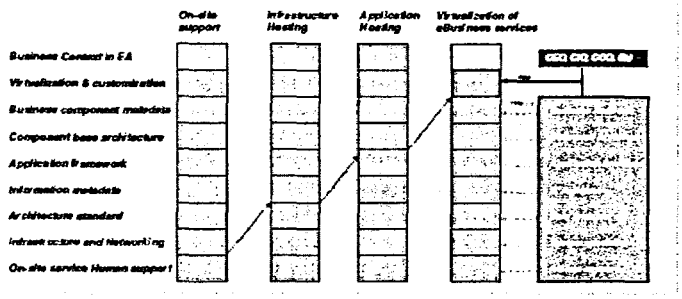


Figure 1. Service virtualization and bottom-up approach

In recent literature, a great deal has been written about achieving interoperability among different metadata schemas. A methodological analysis of interoperability focusing on knowledge organization systems (KOS) was presented in a previous article by the authors (Zeng & Chan, 2004). This article analyzes some of the methods currently used to achieve interoperability in a broader context – that is, among different metadata schemas and applications.

Aggregated services and common components should not be tightly coupled; instead, their dependency and interconnectivity should be based at a parameter and metadata level and their design and interfacing specifications should be fully visible for future development. The reason for this is that the more significant role these skeleton elements play, the higher the complexity of interconnectivity, business criticality and access frequency, which can result in significant impact if a defect is detected too late. In many instances, a defect is a kind of binding between components instead of a coordinate from aggregation.

A service aggregation needs to be implemented in an empirical rule-base in the same way that a phrase needs words, grammar, semantics and a certain level of common recognition. It should not be allowed to be handled for individual needs differently or randomly; this is the reason it should be designed and maintained at the enterprise level as a single view of enterprise solution.

The micro process starts with the mechanism for how the component is designed: this involves taking the business semantic of the macro level with coherence of enterprise interoperability between each layer. Before the design of the service component, we take direction from the business component by using a top-down approach to derive the component design while at the same time streamlining the alignment between service components and physical components by using both top-down and bottom-up approaches. Most of the bottom-up approaches are a componentization process working from existing information components, business objects, codes, system metadata or database access protocols. Once we construct the alignment between business, service and information and have the service component design and interconnectivity completed, we actively work on service layer transformation. The transformation is shown in Figure 2.

The reason this step is crucial is that the goal of vertical enterprise integration is to create interoperability between layers from both micro and macro perspectives, with both business and information layers working as a foundation to support the formation of the service layer which provides the actual services. Once the service component has completed its design with alignment to business and information, the focus can be shifted to micro interference within the macro view. In our previous research on metadata interoperability we revealed, in relation to both political and rational models, how the components are derived through the validation process and how the strategy of component implementation achieves layer interoperability. In this paper we extend our research in component design and technical implementation for both internal and interface component factors [1].

Micro process in component transformation

As the objective is to build rule-based enterprise aggregated services, we introduce the "state" concept to represent the intermediate "enterprise status" between transformation and post-transformation. The state concept in our micro process transformation is not merely dealing with a component or a service, but with the service layer and its coherence of enterprise interoperability of business and information. The transformation therefore requires an event to incur the service with the intelligence to either predict the next service or to verify that the called service is recognized. We use the state concept discussed above to mediate the business, service event and information in the "time" dimension.

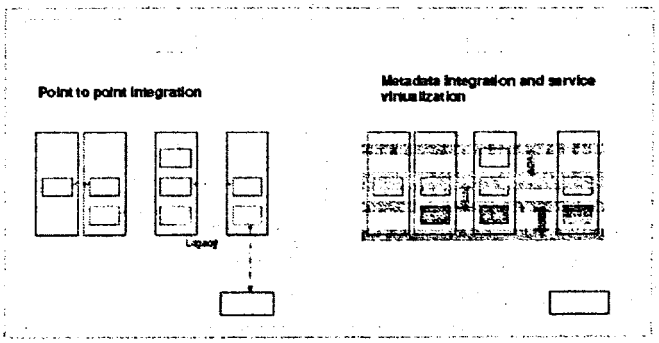


Figure 2. Project architecture benchmarking

The state of the service layer and the business-service-information coherence is changed once we trigger an event and the service is successfully executed. The alphabet read in automata is leveraged to service component trigger and execution while the state change is leveraged to transformation within the service layer.

When the service event Φi ($i <> 1$) is read and triggers the service component, we express the state transformation as follows:

$\varepsilon^{i} : r_{i-1} \times \Phi_i \longrightarrow r_i$, we further derive the transformation as follows:

$\varepsilon^{i+1} = (r_i) \times \Phi_{i+1} \longrightarrow r_{i+1} = (r_{i-1} \times \Phi_i) \times \Phi_{i+1} \longrightarrow r_{i+1}$

which is in equilibrium to $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \longrightarrow r_{i+1}$, and derives $\varepsilon^{nm} : r_n \times (\Phi_{n+1} \times \Phi_{n+2} \ldots\ldots \times \Phi_m) \longrightarrow r_{m\ (n<m)}$, which we call "semantic event aggregator."

This means, the event "aggregates" the service component by using semantic metadata. When we eliminate number of event, we already simplify and build a potential main route for a particular service and bypass intermediate steps by using components' interconnectivity and service plan to transform service into a solid enterprise skeleton. In any component context a formal interface contract for the invocation syntax is important. The semantics issue (the meaning of parameters and so forth) has to be solved as well (domain modeling). This is crucial to any business-to-business (B2B) and dynamic invocation scenario. Such scenarios are cornerstones of the service component vision of being flexible and agile in response to the new business needs in a world of mergers and acquisitions, business transformation, globalization and other relevant factors [2].

The string $\Phi (r_n , r_m) = \Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m$ is a path from r_n to r_m, we call the "service route" and $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \longrightarrow r_{i+1}$ a "composite service" between r_{i-1} and r_{i+1}, we use the symbol $\Phi_{i,\ i+1}$ to denote a composite service.

When frequency $F(\Phi_{i,\ i+1}) = F(\Phi_i \times \Phi_{i+1})$ is higher than threshold σ, we aggregate the concatenated alphabet (part of the string) $\Phi_i \times \Phi_{i+1}$ into $\Phi_{i,\ i+1}$, this implies that the original state r_i between r_{i-1} and r_{i+1} will no longer exist for this composite service.

When the business architect checks the result from semantic event aggregator - ε^m which is more an upper-layer use case scenario, service route - $\Phi_{i,\ i+1}$ which is in the service layer and its frequency - $F(\Phi_{i,\ i+1})$, if $F(\Phi_{i,\ i+1})$ is much higher than threshold σ, then from the business perspective the event aggregator ε^m either consists of many composite services or is "business critical." Business critical means the aggregation service is key within the business and is crucial in daily operation from the service perspective, and that fine tuning is feasible from a multi-fine-grained boundary to a coarse-grained service.

We call this the "main route" of the enterprise skeleton; in other words this is the highway between A and B. Virtual coarse-grained service eliminates the boundary and leverages many fine-grained components which can be easily configured by adding or modifying the component dynamically to achieve the broader coverage of main route services. For this reason the main route is one of crucial services in enterprise integration. One example is in the telecommunication industry that incorporates an open service account, an open billing account and provisioning options for a new customer. This constitutes a main route for "new customer service" coarse-grained service, though it is aligned with fine-grained components to support

customer preference, service criteria and billing options which are branches within this main route.

In service route $\varepsilon^{nm}: r_n \times (\Phi_{n+1} \times \Phi_{n+2} \ldots\ldots \times \Phi_m) \rightarrow r_{m,\,(n<m)}$ if these exist in a composite service Φ_i that r_i is omitted ($n < i < m$). Within this framework individual communication protocols, data manipulation languages and data are represented in a generic manner while preserving their ontological variety; this process indicates how a mediation infrastructure can be built using canonical wrappers, operational requirements of integration and layered architecture for component design [3].

Micro interoperability in the component-based design

The importance of determining the common components and service aggregation is in building up an enterprise skeleton of vertical interoperability between business, service and information. This task is crucial, as in the business domain analysis stage, the impact area and scope of business components are minimized in decision making. Additionally, the business component is deconstructed into service components and physical elements for top-down alignment while the common component is not simply a symbolic service but instead requires a bottom-up approach to work on common areas; this serves to fine tune the common components and interfaces from the perspective of enterprise integrity and performance. The service aggregation is also important as the top-down deconstruction followed by bottom-up composition forms the skeleton of vertical interoperability and potential component-based implementation of the service process.

The iteration through each use case will be tracked for its path and service statistics so that once we parse all use cases we should be able to see the frequency of use of each service as well as the path and dependency between services. Where there is a high frequency of services we use the term "common services" and focus on the potential for reuse and minimization of the integration interface. In the case of service paths in which several services have a common sequence and appear in use cases a certain number of times past the threshold, the services can be aggregated to a composite service or potential process. This needs to be designed at an enterprise level so that each organization has the same concept of and specification for this common service. Service aggregation means a composite component consisting of at least two components that form a unique sequence of execution order with a common interface within their boundaries; this composite component will be treated as a coarse-grained service in provisioning the upward aggregation of services. The pre-built foundation of K-B-S-C alignment increases portfolio provision agility for each strategic decision K, so that each K (the "active map" consisting of the selected business components from the business component model) gives a clear picture of which business components should be the focus in deriving service elements.

Once the vertical skeleton and service extension are built, the interoperability reflected from the business layer has the capability of integrating cross-organization and cross-geographical constraints, as fully transparent services become possible if each system or organization follows an industry standard by using metadata strategy and an industry framework. An example of this would be a case in which several key industry vendors have built up an industry framework for service providers in that particular industry to follow, starting from business domain analysis and working through to component deployment. Each service provider would thus benefit from the

industry-common services and the common interface, particularly those involving merger and acquire cases (B2B, B2C). Any business activity becomes easier when the various parties involved have a common understanding and a consensus on business terms, service definitions, interfaces and component specification, particularly if these become transparent and common practices. The important concept here is that once the metadata of business, service and information constructs an industry standard, the organization's vertical skeleton has the opportunity to integrate across multiple organizations for an easier global practice; this is a long-term industry objective which we call service virtualization in a fully transparent environment.

The foundation of this skeleton is achieving interconnectivity between service components by using a common component and an aggregated component. We further develop the service virtualization on top of this interoperability skeleton, which we call enterprise vertical integration, in extending the integration across industry and geographical areas. This is the virtualization service in a horizontal direction; once the industries of different regions build up their semantic interoperability in three metadata repositories (business, service and information), they can synchronize their business, operation and strategy even if they have their local implementation extended from the interoperability skeleton. This is what we call service virtualization; we predict that virtualization of services will become a priority in industry as semantic interoperability is in high demand to solve the issues in both vertical interoperability and horizontal interconnectivity.

Conclusion

We introduced the concept of event aggregator, which means that the event aggregates the service components by using semantic metadata. We found that within service components there is a referential integrity between components, and the relationship can be modeled and quantified by using the concepts of aggregation and automata. The analysis is crucial to any business-to-business (B2B) and dynamic invocation scenario as such scenarios are cornerstones of the service component vision of being flexible and agile in response to the new business needs in a world of mergers and acquisitions, business transformation, globalization and other relevant factors. In transformation of service aggregation we explored the possibility of applying automata theory in collaboration with service aggregation. In enterprise service provisioning and virtualization, we constructed the skeleton of enterprise service with leverage to each layer and the interoperability of the layers.

Service virtualization is a process of leveraging enterprise integration strategy within a specific geographical region or across regions. This can be an industry standard scenario such as a credit card transaction in which the service, geography and systems are fully transparent to both cardholders and merchants. A full virtualization environment needs to be built on top of the foundation of component design and service transformation process in order to apply the vertical enterprise integration approach to this final stage, which is the goal of our research work. The standardization of metadata technology and common services and components are all requirements in this stage.

References

1. Jie Lu, Guangquan Zhang & Fengjie Wu, (2005), "Web-based Multi-Criteria Group Decision Support System with Linguistic Term Processing Function", IEEE Intelligent Informatics Bulletin June 2005 Vol.5 No.1
2. O. Zimmermann, P. Krogdahl, C. Gee, (2004), "Elements of Service-Oriented Analysis and Design", IBM Academy
3. Sergey Melnik, (1999), "Generic Interoperability Framework", Distributed and Parallel Databases Journal 1999

C6: Concept of service provisioning and realization

ISKE-2007

第二届智能系统与知识工程国际会议

Conference on Intelligent Systems. and Knowledge Engineering

Oct 15-16, 2007, Chengdu, China

第二届智能系统与知识工程国际会议（ISKE2007）

第二届智能系统与知识工程国际会议（ISKE2007）

成都，中国，2007年10月15-16日（http://iske2007.swjtu.edu.cn/）

第二届智能系统与知识工程国际会议（ISKE2007）将在探讨与智能系统和知识工程相关的各个领域的最新研究和进展：

- 在2006-2007中国——比利时国际合作项目："面向数据挖掘和信息处理的智能系统的方法与应用"框架下，讨论解决知识工程问题所采用的智能系统；
- 促进中国和国际研究机构和大学的共同研究，搭建计算智能和知识系统之间的桥梁；

Chengdu, 2007-07-25

Dear Authors,

We are pleased to inform you that your paper:

ID	007
TITLE	Concept of service provisioning and realization
AUTHORS	Raymond Wu

has been accepted as a full paper for the final program of 2007 International Conference on Intelligent Systems and Knowledge Engineering (ISKE2007, Chengdu, China, October 15-16). Congratulations!

The review process was extremely selective as we have received over 700 submissions. ISKE2007 will have a total of about 300 papers in the final program. The complete review process has not been finished yet, but your paper has been included in the first batch of accepted papers!

Thank you very much for your support to ISKE2007. We are looking forward to welcoming you in Chengdu!

Yours sincerely,

Tianrui Li, Jie Lu, Etienne E. Kerre

ISKE07 Program Chairs (http://iske2007.swjtu.edu.cn)

Concept of service provisioning and realization

Raymond Wu
IBM Software Group
raymond@de.ibm.com

Abstract

Service transformation is one of the big challenges in this century from a scale and complexity perspective. The main objective is to increase service levels, enhancing both customer satisfaction and productivity. To reach this goal requires multi-dimensional approaches from business innovation, service strategy, solution architecture, implementation and integration. Therefore, alignment, coherence and a roadmap are very important. In this research paper we introduce conceptual service provisioning with extension to interconnectivity as the service foundation. We further explore service provisioning based on the concept we introduced and realization based on service design pattern foundation.

Introduction

The concept of service provisioning requires service identification, decision, validation, analysis and design. Our previous research within enterprise integration has been divided into four areas: decision, analysis, design and virtualization. Within the life cycle, political-model decision-making and rational-model analysis were applied to the key areas in which the components of business, service and physical elements are selected and validated in provisioning a particular business context goal or KPI within the project scope of the enterprise integration dimension.

In the research on metadata interoperability we revealed, in relation to both political and rational models, how the components are derived through the validation process and how the strategy of component implementation achieves layer interoperability. In this paper we extend our research in service concept and service provisioning, and define the service ontology (Ψ) as follows:

In a full service interoperability within micro and macro processes:

Demand = Supply = Provisioning, means: $\Psi = \Sigma\,(\Psi(d_i) \times \Psi(s_i) \times \Psi(p_i))$

This implies that service provisioning is implemented through the micro process and component alignment between service, demand and supply, so each required element, in either new or enhanced form, can be narrowed down to one or more mapped components which are consolidated by the object, symbol and service concept. The realization of this top-down vertical integration approach requires robust middleware to support the inter-communication between component-and-component and component-and-repository. In other words, the concept needs physical support from an integration service bus such as the middleware in runtime services.

Figure 1 shows the micro interoperability within service ontology. Common service and information is a top-down approach which follows the nature of the business implementation direction. From the technical perspective the idea was to make integration easier by using the "virtualization" concept. From the organization perspective this enhances the flexibility of service provisioning and the agility between organization and service, thus increasing the interoperability across organizations.

Literature review

Oberle, Staab and Eberhart (2006), in their Kaon server design, show several potential sources of semantic metadata. These sources provide input for the framework regarding the semantic management of software components; this includes deployment descriptors, web and application server configuration files, source code information or annotations and database metadata. The component metadata collector parses this information and automatically integrates it into the ontology as semantic metadata (instances). The inference engine is embedded in the application server and applies the ontology and semantic metadata for reasoning and querying. The application server's core functionality and the administration console both use the reasoning capability. The latter allows the developer to browse and query the ontology at runtime so that configurations may be queried or checked to avoid inconsistent system configurations during development and at runtime Semantic mediation between object and symbol has been with us for quite some time; in Greek philosophy, semantic ontology was developed in semiotics to include all of linguistics as a special case. Aristotle distinguished objects, the words that refer to them, and the corresponding experiences in the conceptual experience (Sowa, 2000). The triangle of integration ontology consists of object, symbol and concept. In our conventional design we emphasized the use of a symbol to describe the object, and attempted to synchronize for the moving object.

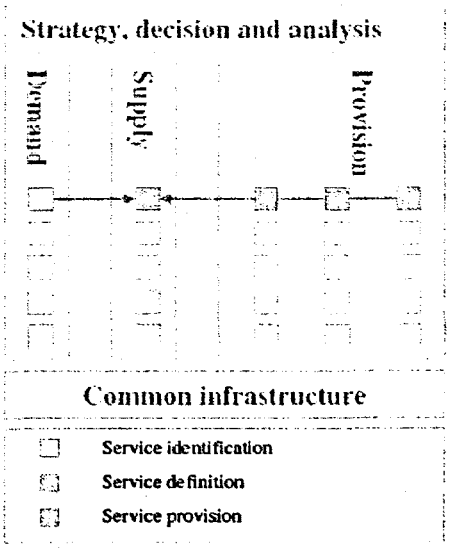


Figure 1. Demand, supply and provision

In many large scale enterprises, architecture such as SOA (service oriented architecture), MDA (model driven architecture) and EDA (event driven architecture) use many conceptual metadata to mediate between the "dynamic" and "static."

Service concept

In an advanced CRM (customer relationship management) system, we assume that all the customer master data needs to be static and consistent; this is due to information and process not having sufficient "speed" to synchronize with changes in market, customer preference and industry trends if the concept does not support dimensional strategy.

What we need is a service concept mediation between object and symbol. Figure 2 demonstrates this mediation. A racing car consists of an ontology of three parts: object (physical car), symbol (description) and concept (a conceptual car and racing). Our focus is on concept instead of object or symbol, as the object in this case is the car itself and does not contribute to the objective, and the symbol notation contributes no semantic information.

Concepts change from time to time; in the case of a racing car we take the statistics information from the symbol which points to the object (car), then we aggregate the symbol using such data as analytical information about the behavior of this car and finally input the metadata into the concept layer; in this way more strategic governance will be produced from the concept to cope with a changing environment.

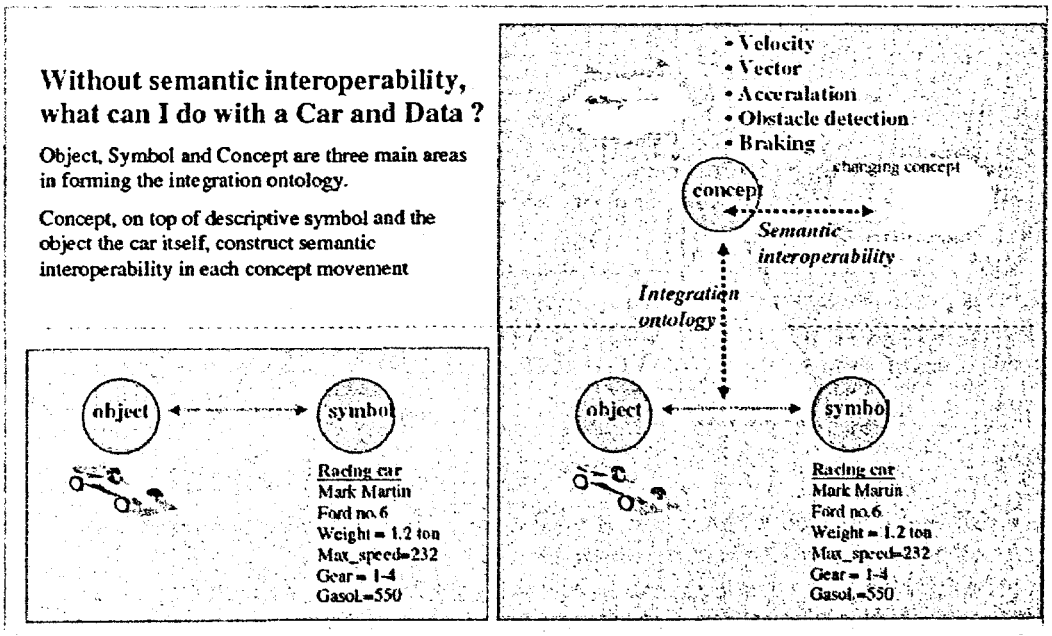


Figure 2. Service concept mediation in service ontology

Let 'τ' denote the object, the physical car

'χ' denote characteristics of the car (weight, engine, etc.) - the symbol

'v' denote velocity

'α' denote acceleration

'ω' denote direction

'β' denote braking

'δ' denote distance between car and obstacle

'Ψ' denote the ontology interoperability between v, α and ω

Let the concept of "win in racing" be constructed by semantic integration $\Gamma = \Psi (v, \alpha, \omega, \beta, \delta)$ and $\Psi^1, \Psi^2 \ldots \Psi^n$ denote the change of strategy in coping with environment, trend, competition, KPI (key performance identifiers) or market need (win). In the case of the racing car this can be the competition strategy and environment.

Concept Γ in a moving environment produces the semantic interoperability in each change; we use $\tilde{O} = \Gamma (\Psi, \tau, \chi)$ to denote the ontology of enterprise integration. This clearly shows that the ontology $\tilde{O}$ built on top of the interoperability between object, symbol and concept needs recursive interference to adjust its interoperability $\Gamma^* = \Gamma + \Delta\Gamma$

The condition in producing $\Delta\Gamma$ is based on $\tilde{O} = \Gamma (\Psi, \tau, \chi)$, instead of $\Gamma = \Psi (v, \alpha, \omega, \beta, \delta)$ In other words, our assumptions are that design mode $\tilde{O} = \Gamma (\Psi, \tau, \chi)$ is the origin in moving execution mode from Ψ^{i-1} to Ψ^i, concept Γ mediates "racing" and "car" in competition, semantic interoperability creates the dimensional strategy in mediating object and symbol, and the componentization is the foundation in this ontology.

The design of service components follows our previous strategy by using business semantic injection to take the context into the semantic and micro processes; this means each component has its contract-level responsibility in provisioning a subset of service. This interface helps the coordination between components and metadata to build the foundation of interoperability within the service layer and between layers. As we are still at a pioneer stage in service provisioning and aggregation, we understand this cannot be accomplished within a short period. Further research efforts will be undertaken following the current work [3].

Service mediation

As service provision requires service mediation to serve as foundation. Now this study need to shed light on the realization for how mediaters support the concept this paper proposes. D. Roman et al. in their "Web Service Modeling Ontology (WSMO)" state that semantic services need to meet the following requirements: web compliance, ontological base, strict decoupling, centrality of mediation, ontological role separation, and execution semantics [4]. These requirements were addressed in our research and the strategy is identical to that which we proposed. Basically, mediation patterns manipulate messages

in-flight on the bus by requests or events.

According to IBM service mediation, there are several basic patterns for mediations; the following more complex patterns can be built by combining the simple patterns:

Protocol switch: Enables service requesters to dispatch their messages using a variety of interaction protocols or APIs such as SOAP/HTTP, JMS and the MQ Integrator (MQI).

Transform: Translates the message payload (content) from the requester's schema to the provider's schema. May include enveloping, de-enveloping or encryption.

Enrich: Augments the message payload by adding information from external data sources such as customization parameters defined by the mediation or from database queries.

Route: Changes the route of a message, selecting among service providers that support the requester's intent. Selection criteria can include message content and context as well as a target's capabilities.

Distribute: Distributes the message to a set of interested parties and is usually driven by subscriber interest profiles.

Monitor: Observes messages as they pass through the mediation unchanged. Can be used to monitor service levels, assist in problem determination or meter usage for subsequent billing to users.

Correlate: Derives complex events from message or event streams. Includes rules for pattern identification and rules that react to pattern discovery: for example, by generating a complex event derived from the content of the triggering event stream.

Mediations can be explicitly configured within a solution. As well as being attributes of the services, policies can be set by the solution administrator for a single interaction or set of interactions [5]. Once the vertical skeleton and service extension are built, the interoperability reflected from the business layer has the capability of integrating cross-organization and cross-geographical constraints, as fully transparent services become possible if each system or organization follows an industry standard by using metadata strategy and an industry framework. An example of this would be a case in which several key industry vendors have built up an industry framework for service providers in that particular industry to follow, starting from business domain analysis and working through to component deployment. Each service provider would thus benefit from the industry-common services and the common interface, particularly those involving merger and acquire cases (B2B, B2C). Any business activity becomes easier when the various parties involved have a common understanding and a consensus on business terms, service definitions, interfaces and component specification, particularly if these become transparent and common practices.

The foundation of service interoperability is achieved by interconnectivity between service components by using strategy of common component and an aggregated component. We further develop the service virtualization on top of this interoperability skeleton, which we call enterprise vertical integration, in extending the integration across industry and

geographical areas. This is the virtualization service in a horizontal direction; once the industries of different regions build up their semantic interoperability in three metadata repositories (business, service and information), they can synchronize their business, operation and strategy even if they have their local implementation extended from the interoperability skeleton.

In enterprise service provisioning and virtualization, we constructed the skeleton of enterprise service with leverage to each layer and the interoperability of the layers. Service virtualization is a process of leveraging enterprise integration strategy within a specific geographical region or across regions. This can be an industry standard scenario such as a credit card transaction in which the service, geography and systems are fully transparent to both cardholders and merchants. A full virtualization environment needs to be built on top of the foundation of component design and service transformation process in order to apply the vertical enterprise integration approach to this final stage, which is the goal of our research work. The standardization of metadata technology and common services and components are all requirements in this stage.

Conclusions

In the new service provisioning industry, the object and the symbol – the description of the object – have been the focus of interest; however, the service concept behind the object and the symbol, is the real key in provisioning the demanding service and maintaining the consistency between object, symbol and service. The service concept mediates the object and the symbol within the service ontology as concepts change from time to time. We use the statistics information from the symbol which pointed to the object, we then aggregate the symbol using such data as analytical information about the behavior of the object and finally input the metadata into the concept layer; in this way more strategic governance will be produced from concept to cope with a changing environment. In service realization, web compliance, ontology, decoupling, centrality and semantics are keys in building the patterns. Basically, mediation patterns manipulate messages in-flight on the bus by requests or events.

References

1. D. Oberle, S. Staab and A. Eberhart, (2006), "Semantic Management of Distributed Web Applications", IEEE Distributed Systems Online, vol. 7, no. 4, 2006, art. no. 0605-o5001

2. John F. Sowa, (2000), "Ontology, Metadata, and Semiotics", ICCS 2000 AI #1867 Springer-Verlag pp. 55-81

3. R. Barcia and J. Brent, (2005), "Building SOA solutions with the Service Component Architecture", IBM Research

4. D Roman et al, (2005), "Web Service Modeling Ontology (WSMO)", European Commission under the projects DIP 2005

5. B. Hutchison and M. Schmidt and D. Wolfson, (2005), "SOA programming model for implementing Web services", IBM Research 2005

C7

Architecture Rationalization
Part I – Introduction & Action Plan

Version 1.1
January 10, 2006

Prepared by: Raymond Wu
Chief Technology Officer
EDS Greater China

Document Status:	APAC Regional Compliance
Version Number:	1.1
Date:	01/10/2006
Status:	Peer Reviewed

Document Revision History

Version	Authorised Date	Reason for Change	Author Name	Responsibility
0.1	December 15, 2005	Basic information	Raymond Wu	Chief Technology Officer
1.0	December 30, 2005	Initial draft	Raymond Wu	
1.1	January 10, 2006	Reviewed	Raymond Wu	

EDS is a registered mark of Electronic Data Systems Corporation. EDS is an equal opportunity employer, m/f/v/d.
Copyright © Electronic Data Systems Corporation. All rights reserved

Table of Contents

1. Introduction
   - 1.1 Summary
   - 1.2 Rationalization requirements and assessment for GC&K development infrastructure
   - 1.3 Why Rationalization
   - 1.4 The global IT pain
   - 1.5 Top root issues need IT rationalization
   - 1.6 IT Rationalization approach
2. Rationalization action
   - 2.1 Infrastructure Modernization
     - 2.1.1 Version upgrade (immediate upgrade)
     - 2.1.2 EDS portfolio platform (long term plan)
     - *2.1.3 Core Infrastructure & Network*
     - *2.1.4 Servers & workstation (desktop/laptop)*
     - *2.1.5 Remote Access*
     - 2.1.6 COE Software Standards
     - 2.1.7 Project based transformation long term plan
     - 2.1.8 Standardize file transfer & messaging in multi-tiers systems including mainframe, application, gateway and web server
   - 2.2 Estimated Trend and Proposal for future Infrastructure development in EDS GC&K
   - 2.3 Tool deployment and design tools
     - 2.3.1 Borland Tool
     - 2.3.2 Topology recommendation for Starteam logical partition
     - *2.3.3 Application and Design tools*
     - 2.3.4 Development and Application software rationalization
     - *2.3.5 Deviation Process*
     - *2.3.6 Office Automation and Messaging Applications*
     - *2.3.7 Connectivity SoftwareJava Remote Method Invocation Telnet*
     - 2.3.8 Internal Administrative Applications
     - 2.3.9 Project Management
     - 2.3.10 Software Development and Maintenance
     - 2.3.11 Requirements Management
     - 2.3.12 Estimation/Metrics
     - 2.3.13 Software Engineering and Visual Modelling Tools
     - 2.3.14 Knowledge Management
     - 2.3.15 Integrated Development Environment
     - 2.3.16 Cobol Code Generation
     - 2.3.17 Defect and Issue Tracking
     - 2.3.18 Workflow Management
     - 2.3.19 Configuration Management
     - 2.3.20 WWW Development and HTML Help
     - 2.3.21 Database Development
     - 2.3.22 Testing
     - 2.3.23 Reporting
     - 2.3.24 Software Distribution
     - 2.3.25 Operating Systems
     - 2.3.26 Programming Languages
     - 2.3.27 Protocols, Middleware
   - 2.4 Testing key items
   - 2.5 Estimation key items
3. Corporative strategy in rationalization
   - 3.1 Strategic rationalization in eliminating dependency case study - Workflow Java engine to replace Lotus Notes
   - 3.2 Multi-environment application rationalization
   - 3.3 Project rationalization - Governance of project architecture in high complexity development
   - 3.4 Re-engineer the well-defined interface between EDS and client
   - 3.5 Foundation rationalization
4. Application Architecture pre-build foundation
   - 4.1 Summary of Architectural Standards and Foundation
     - 4.1.1 Application Platforms
     - 4.1.2 Application Design Models
     - 4.1.3 Technical Platforms
     - 4.1.4 Tool and Application Development Guidelines
     - *4.1.5 Global secure network*
     - 4.1.6 Future-trend - EDS Pre-build architecture foundation
     - 4.1.7 Future-trend - Collaboration Platform and "V" model
     - 4.1.8 Future trend - Language-oriented programming with domain-specific language
   - 4.2 Agile Enterprise Platform modernization
   - 4.3 Identify and define the architecture pre-build components
   - 4.4 Interface in Integration - To provide a well-defined interface for client and 3rd party vendor

Summary

Rationalization actions and assessment

1. **Infrastructure Modernization**
   - Version upgrade (i.e. Mainframe OS/390 upgrade to zOS, Oracle 7 to 8i...)
   - Standardize file transfer & messaging in multi-tiers systems including mainframe, application, gateway and web server
   - Propose EDS trend and transformation in long term plan
2. **Tool deployment, Application and Design tools**
   - Tools Rationalization - Borland, Endevor & Mercury tools
   - Programming standard and development tools
   - Application related and Design Tools
   - Testing and Estimation key items
3. **Corporative strategy in Rationalization**
   - Strategic Rationalization - Eliminating competitor dependency W/F case
   - Rationalization principles in Multi-environment application development
   - Project Rationalization - Governance of complex project architecture
   - Client Interface Rationalization - Re-engineer the well-defined interface
   - Foundation Rationalization - Re-engineer EDS architecture foundation
4. **Agile and Integration for pre-build foundation rationalization**
   - Pre-build components - Identify and define the agile architecture
   - Integration - To provide a well-defined interface for client & 3rd party vendor
   - Apps pre-build foundation will be increased from 30-50% to 70% including common component and architecture framework
5. **Service Level Review**
   - Current SLA review
   - To achieve the SLA by application rationalization

The critical issue we are facing are;

1. The systems were implemented by different method and techniques cause the diversity & heterogeneous and difficult in establishing a standard and common platform & tools
2. Customization option is much higher than configuration cases cause system loss it's agility
3. Lack of global & enterprise architecture view, and inefficient interfacing in multi-location (i.e. multi-language & multi-currency) support
4. Lack of unique entry point to provide single sign-on (SSO) to facilitate robotic single entry and security control
5. Limited work flow, alert and exception handling in system level
6. Limited on-line query and business intelligent features
7. Weak in API functionality to perform system integration
8. Too much cost & effort were spent in system maintenance

Top root issues need IT rationalization

These root issues are strategic level, usually they came from lack of a common view and need be rationalized from development environment to streamline the design.

Item	Issue	Description	Resolution in rationalization
1	**Misalignment of IT activity to corporate strategy and project architecture**	The lack of a clearly communicated business strategy creates an environment in which efforts are tactically focused on immediate business needs without regard for the future long-term direction of the company	Collaborative design and enterprise view is required in development environment rationalization
2	**Lack of effective business or IT governance**	Without effective business and IT governance, lower-value service requests receive the same priority attention and resource allocation that higher value service requests receive	System implementation need an overview from corporate level to prevent isolation and non-standard implementation
3	**Proliferation of applications**	Without the necessary controls to ensure that each technical investment is justified in terms of business value and adheres to the technical standards, Client application portfolio will continue to grow larger, more complex and more expensive	Keep a close look at ROI so each investment need be business feasible
4	**Duplication of functionality, data systems and support infrastructure**	Many times individual business units have the ability to select the technical means they feel best support their needs. This unfortunately creates a situation where redundant capabilities, data and infrastructure are purchased, duplicating support needs and cost	Enterprise view, consolidation and integration in development environment rationalization
5	**Inappropriate enhancement and use of systems**	Users have focused their technical resources on tactical system enhancements that have resulted in modifications that meet immediate needs, often at the expense of the longer term direction or value.	A view of priority and roadmap need be provided at overall enterprise level to prevent resource misuse or mis-focus
6	**Siloed Operations**	Too often individual business units an organization operate in a very autonomous, siloed manner, limiting the leveragability of technical solutions, data and infrastructure. The result of this situation is a complex support needs and redundant cost structure.	Create visibility & registry for each technical implementation, so a server, infra. component need be validated for integration at enterprise level
7	**Fragmented information architecture**	Too often the acquisition, storage, manipulation, and use of data is done disparately across the organization, resulting in data inconsistencies, redundant data purchases, duplicated support structures, and excess cost.	Enterprise view and sharing infrastructure is required in development environment rationalization

IT Rationalization approach

Based on the survey of IT root issues, we plan the IT Rationalization approach as followed:

Application Portfolio Rationalization

- Eliminate low value business applications and associated support costs
- Consolidate redundant applications, functionality and support
- Quantify the savings that could be achieved by moving to off-shore support

Infrastructure Rationalization

- Optimize current assets, leveraging capacity across the various business needs
- Consolidate disparate servers, storage, and support infrastructures across enterprise

IT Spending &Staffing Rationalization

- Assess IT spending and staffing against appropriate industry benchmarks
- Determine areas to reduce IT base spending and redirect to higher value areas

IT Organization Transformation

- Establish consistent processes for support, delivery and new projects
- Define governance model, ensuring clearly defined roles and responsibilities
- Identify alternative organizational models that better align to business needs

IT Project Rationalization

- Prioritize projects across the enterprise to optimize the IT investment portfolio and maximize ROI
- Rationalize to eliminate discretionary or low value initiatives
- Accelerate benefits where possible

IT Sourcing

- Assess external spend on IT products and services
- Centralize vendor negotiations and management
- Consolidate total enterprise spend to gain negotiating leverage with vendors
- Defined complimentary IT partnership model

Selective IT Outsourcing

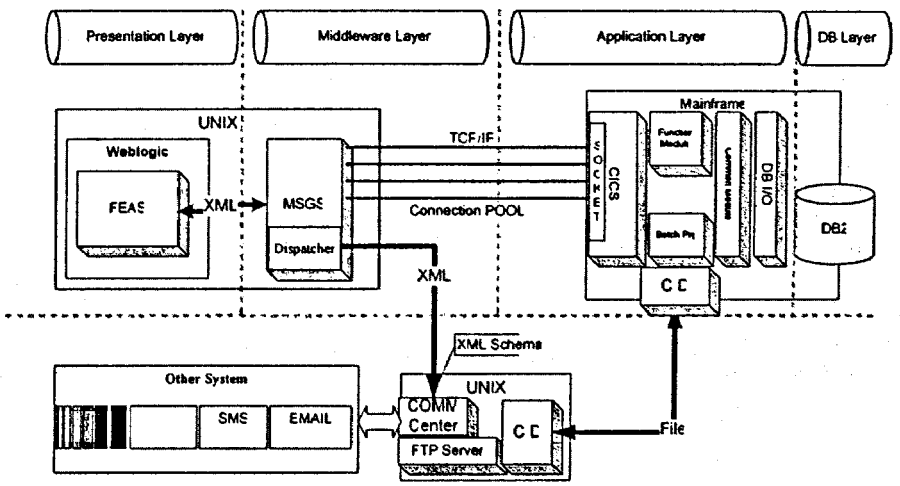
- Applications: Identify legacy or non-core business applications that can be outsourced to allow internal IT resources to be reallocated to higher value efforts
- Infrastructure: Assess internal IT capabilities and spend with industry norms and alternative service (data center, network, telecom, desktop, and help desk services)
- Identify opportunities that improved service levels can be obtained while cost target are achieved

Rationalization action

Standardize file transfer & messaging in multi-tiers systems including mainframe, application, gateway and web server. These is a best practice for layers' communication and message

Transfer, XML is the recommendation for messaging between presentation layer and middleware, and between middleware and application layer.

There is a well-developed middleware gateway between mainframe and unix (presentation web server), the TCP/IP is recommended for communication between common gateway and mainframe, and C:D (connect direct is the option for file transfer between mainframe and unix server)



Trend and Proposal for future development

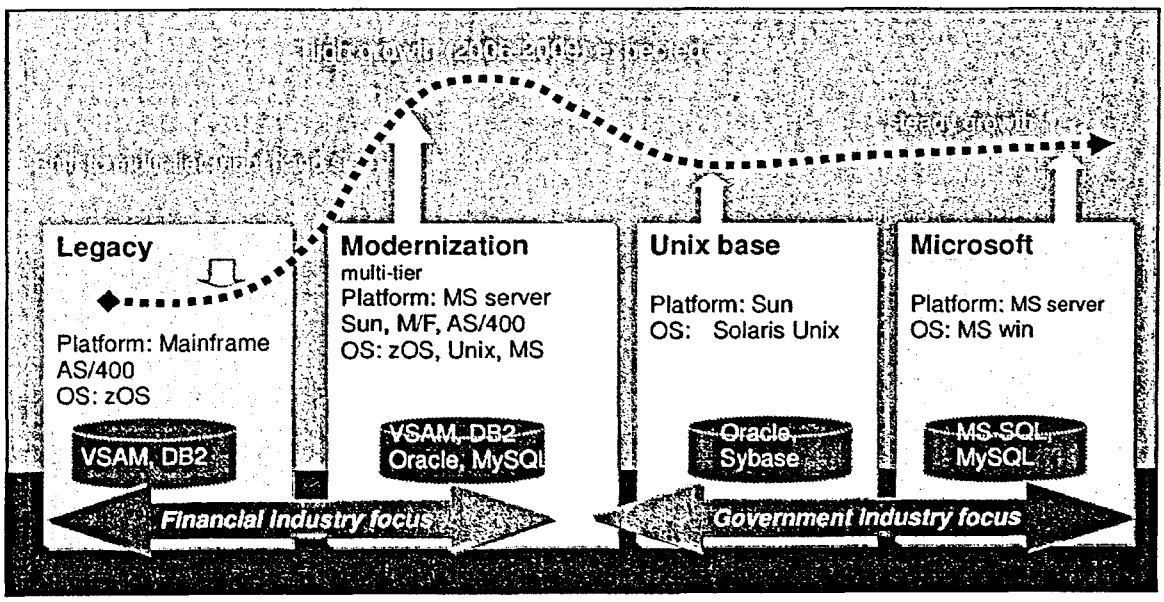
The recommendation of future platform for financial industry is the multi-tiers architecture from modernization, this means shift from legacy systems to multi-tier modern system need consolidation from mainframe, unix server and possible Microsoft server is also needed. The recommendation of Government industry is stay in Microsoft, however some modernization (such as SOA, EDA..) are required.

Deviation Process

The remainder of this section concentrates on listing the approved software products for use. However, it is understood that there will always be unique needs for a given project or team which may require a deviation from the approved products. If a team or individual identifies a need to deviate from the approved products, they are to contact their Technical Leader for approval for such deviation.

Tool deployment and design tools

- Install the production servers with StarTeam and CaliberRM
- Migrate the Source Code
- Install a StarTeam Client onto the Unix Platform for the Unix ProC development
- Seek agreement on a way forward with the change request management system being migrated to StarTeam



CMMI Level 3 compliance and Change Management process

Successful deployment of CM tools is crucial to the success. Review the overall deployment plan to ensure that it is aligned with the priority and objectives of the CMMI L3 plan

- Review the implementation approach of tool on projects, to ensure that the implementation is effective and will meet CMMI CM requirements
- Review Change Management process and procedures in relation to the tool, to ensure that organization and project CM Plans are updated accordingly
- Trained on the tool so that the CM SME will be able to effectively perform baseline verification audits using the tool

Development and Application software rationalization

The following sections describe the software products to be used at the GC&K to develop, test, and control software applications both for internal GC&K use and for external clients. In most cases, the EDS COE, the Asia Pacific Technology Plan, or the Client's software direction will dictate the choice and usage of the product(s). Where there is a conflict, EDS COE and the AP Technology Plan will be used for internal applications and the Client's standards and guidelines for Client projects. It should be noted that in moving to EDS COE tools substantial direct license costs can be eliminated thus improving the overall cost effectiveness of the solution centre.

C8

DB Staging in B2B

Presented to Mitsubishi Nippon and Portal Board (2002/02/24)

This document consists 4 parts –

I. Design Strategy

II. Implementation

III. User Guide

IV. Maintenance

Prepared by: Raymond Wu, Project Manager and Architect
Portal PCC Tokyo

Version 1.0 – Initial draft 2002/02/19
Update History:
Published Date: 2002/03/05
(confidential materials)

Part 1. Design Strategy

This Part is prepared for Design Strategy purpose, it helps us to understand why we need Staging DB to perform Periodical Database Update, what is the design strategy, the infrastructure and system components, how it works to detect the errors and smooth the change, the advantages and cutting-edge technologies. The following paragraph consists of –

a. Reasons of Staging DB (Staging DB の 理由、動機)
b. Components of Staging DB (Staging DB の コンポーネント)
c. Mechanisms of Staging DB (Staging DB のメカニズム)
d. Specification of Staging DB (Staging DB の 明細)

Ia. Reasons of Staging DB

1. 'Sharing and Collaborative' technologies has become the dominant E-Business solutions (「共有、そして共同制作」は支配的な E-business Solution になった。)
2. If the Data Input Process is not well-controlled, the Administrator, the Company and Users will suffer from those messy data change as –
   (データ入力プロセスが上手にコントロールされていない場合、管理者、会社 およびユーザは ---としてそれらから汚すデータ変更を受けるでしょう)
   - Nobody know who did what (誰が何をしたか誰も分からない)
   - How good the data quality (データの質がどれ位良いか)
   - How and where the errors come from (エラーがどのように、またどこから来たか)
   - How to prevent the errors (エラーをどのように防ぐか)
   - Potential risks of database crash (データベースクラッシュを引き起こす可能性のあるリスク)
3. Staging DB is working for the solutions that
   - Maintain the data integrity for users to Query and Download from Production to their local environment (through XML)
     (Staging DB はソリューションのために次のことを動かすことです。)
   - Allow the users to work (Update, Delete and Insert) on the download data and upload it to the private tables after offline data manipulation
     (ユーザがダウンロード．データを更新し、削除し、かつ挿入し、かつデータ操作の後に個人のテーブルにアップロードすることを可能にします。)
   - Use Filter Screen Automation to perform the System Validation and detect the errors (エラーを検知するシステムバリデーションを実行するためにフィルタ．オートメーションを使用します。)

- Use Manual Administrative Validation to Validate each input record (各入力レコードを有効にすることのためにマニュアルの管理上バリデーションを使用)
- Maintain the data integrity, apply the change in private tables and move the data to Staging DB (データ保全を維持し、個人のテーブルから StagingDB に変更を適用します。)
- Transit the Staging DB to Production on a Daily basis, Production DB takes less than 1 minute's system downtime to synchronize with Staging DB (一日単位でプロダクションへの StagingDB を通過します。)
- Verify the status of each batch summary record and it's associated records in each step of Validation, Staging and Transition (確認、Staging および Transition の各ステップで各バッチ要約のステータスおよび そのレコードであることを確認します。)
- Report the Activity Status Summary to the Company (会社に活動ステータスを報告すること)
- Tracking is Available if any Summary is not clear
- Reverse is possible under special request (逆は特別のリクエストの下で可能です。)
- Potential Projects of Global/Local Architecture (グローバルとかローカルアーキテクチャーの潜在的なプロジェクト)
- Special Projects and Customization is available (特別計画とカスタマイゼーションは利用可能です。)

Ib. Components of Staging DB

Infrastructure

- Two identical *'Sun Microsystems'* work as Production and Staging systems with potential Active/Warm-standby value-added architecture
- Each machine runs *'Solaris Unix 5.8'* as it's OS
- Each Unix box runs *'Oracle 8i'* as it's DB engine
- *'Client/Server'* Architecture with parallel server capability
- *'Web'* online accessibility co-exist with *'XML'* local offline process

Front-end Development

- 'EJB, JSP, JAVA, XML and VB' technologies are used as User Interfaces to access database
- 'Event Driven'' concept is used for users to click and initiate an event
- 'Query and Manipulation' concept is used in performing Download and Upload batch processes

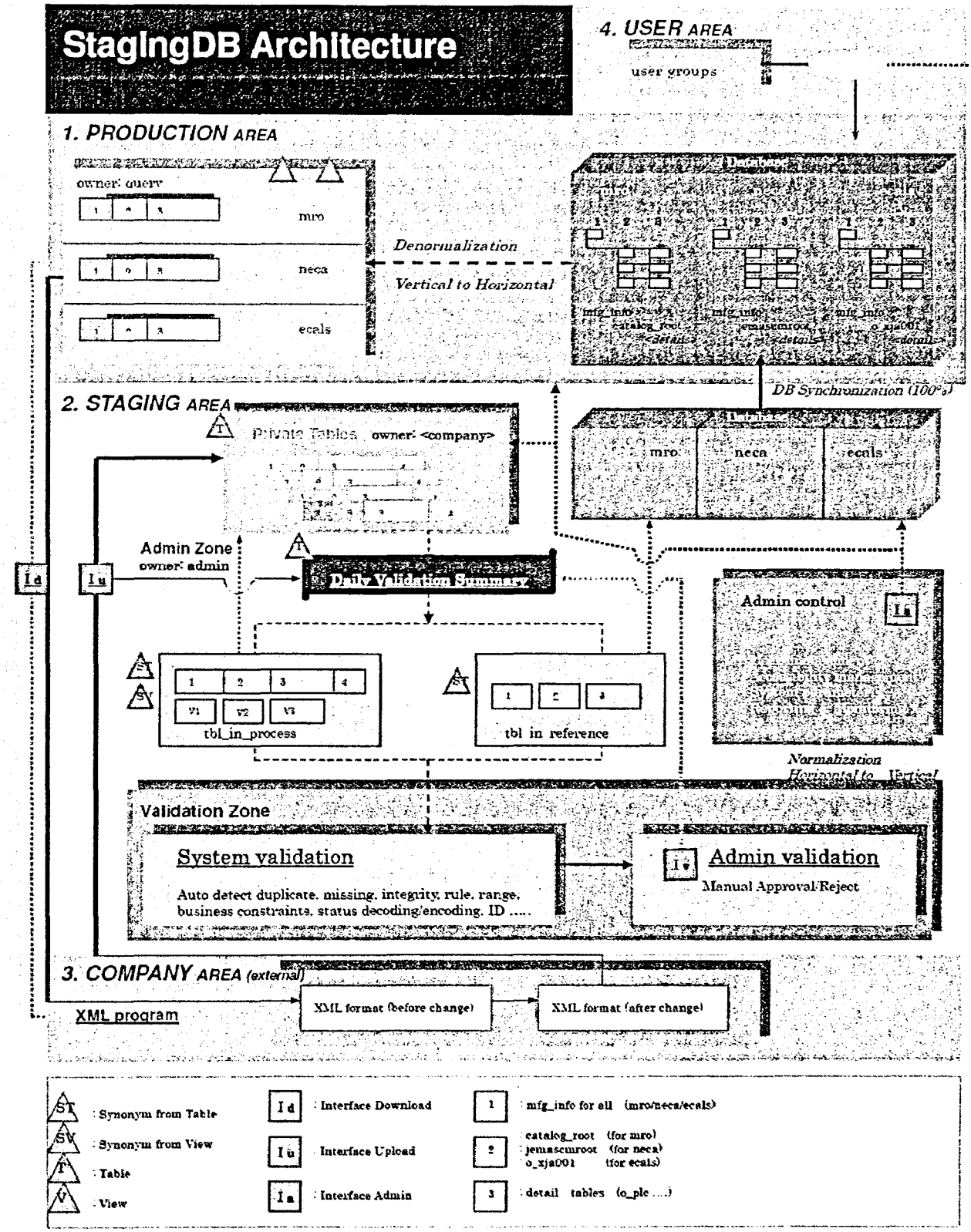
- Interface is separated into 'Query and Input Mode' Query Mode use Hints & Wild Card to enhance query while Input Mode block p/f keys and auto_generate keys in Insert case
- 'Offline Data Manipulation' concept is used to minimize the Networking Traffic
- All the Application and Administration processes are executed through the User Interfaces, users only required 'Login and Click' to operate the process
- 'Mass Data Change' capability makes any single change be applied directly to all selected records
- 'Multi-Language' capability allow users to opt the language preference in both title and data

Database Development

- Table, View and Synonym are used as 'Static database components'
- Trigger, Sequence, View and Synonym are used as 'Dynamic database components'
- SQL and Unix Shell scripts are used as 'Process Languages'
- 'De-normalization' (vertical to horizontal) capability for Query & Download
- 'Normalization' (horizontal back to vertical) capability for data distribution in Staging process
- 'Status Tracking' capability in Validation process by using status code in Summary & Details records
- 'Batch Number' is auto-generated in the Batch Summary & detail records for tracking & integrity
- 'Cross Checking' capability in Validation Process by using Parallel Check and Sequential Check for the duplicate ID and Unique-keys
- 'Auto-generate' ID and Foreign key capability

System Operation

- Use advanced *'DB Synchronization technology'* to synchronize the Production DB from Staging DB Production DB takes less than 1 minute (35-40 seconds) system downtime for the full image
- Use *'Dynamic Reverse Processes'* for the reverse generated SQL/Shell scripts and auto-run the reverse-generated scripts
- Use *'Cron job'* to automate the Batch Processes
- Potential *'Active/Warm-standby architecture'* provide 7 x 24 high redundancy services
- Potential *'Global/Local database architecture'* that each company share the Global information and own it's private dataset


StagingDB Architecture
4. USER AREA
user groups
1. PRODUCTION AREA
owner: query
mro
neca
ecals
Denormalization
Vertical to Horizontal
DB Synchronization (100%)
2. STAGING AREA
Private Tables owner: <company>
mro
neca
ecals
Admin Zone
owner: admin
Id
Iu
Daily Validation Summary
Admin control
Ia
tbl_in_process
tbl_in_reference
Normalization
Validation Zone
System validation
Auto detect duplicate, missing, integrity, rule, range, business constraints, status decoding/encoding, ID .....
Iv
Admin validation
Manual Approval/Reject
3. COMPANY AREA (external)
XML program
XML format (before change)
XML format (after change)
: Synonym from Table
: Synonym from View
: Table
: View
Id : Interface Download
Iu : Interface Upload
Ia : Interface Admin
1 : mfg_info for all (mro/neca/ecals)
2 : catalog_root (for mro)
jemascmroot (for neca)
o_xja001 (for ecals)
3 : detail tables (o_plc ....)

Validation Process (action code: null 'P' ⇨ 'S' ⇨ 'C' ⇨ 'T' ⇨ 'V')

- *During System Validation*: All the Validation Summary are NULL in Action Code. They are changed to 'P' code during System Validation, all the associated records of that Batch Summary record are initially come with Null value in Validation, they are validated against their Action code and receive a value of Validation code after the System Validation.
- *After System Validation*: Only the record with 'S' validation-code and 'S' action code in it's associated Validation-Summary will receive Admin Validation.
- *After Admin Validation*: Only the record with 'A' validation-code and 'C' action code in it's associated Validation-Summary will be applied it's change to Staging DB.
- *After Staging*: Only the record with 'v' validation-code and 'C' action code in it's associated Validation-Summary means record is verified and is ready to be transited to Production DB.
- *After Transition*: Only the record with 'V' validation-code and 'T' action code in
- it's associated Validation-Summary means record is verified and was successfully transmitted to Production DB that finish verification & whole process for that record.

<u>Validation in Summary</u>
Action code:

- Null: not started
- S: finish sys validation
- C: finish adm validation
- T: transition ok
- v finish stgDB vedification
  V finish prodDB vedification

- P: unable to complete S
- Q: unable to complete C or T

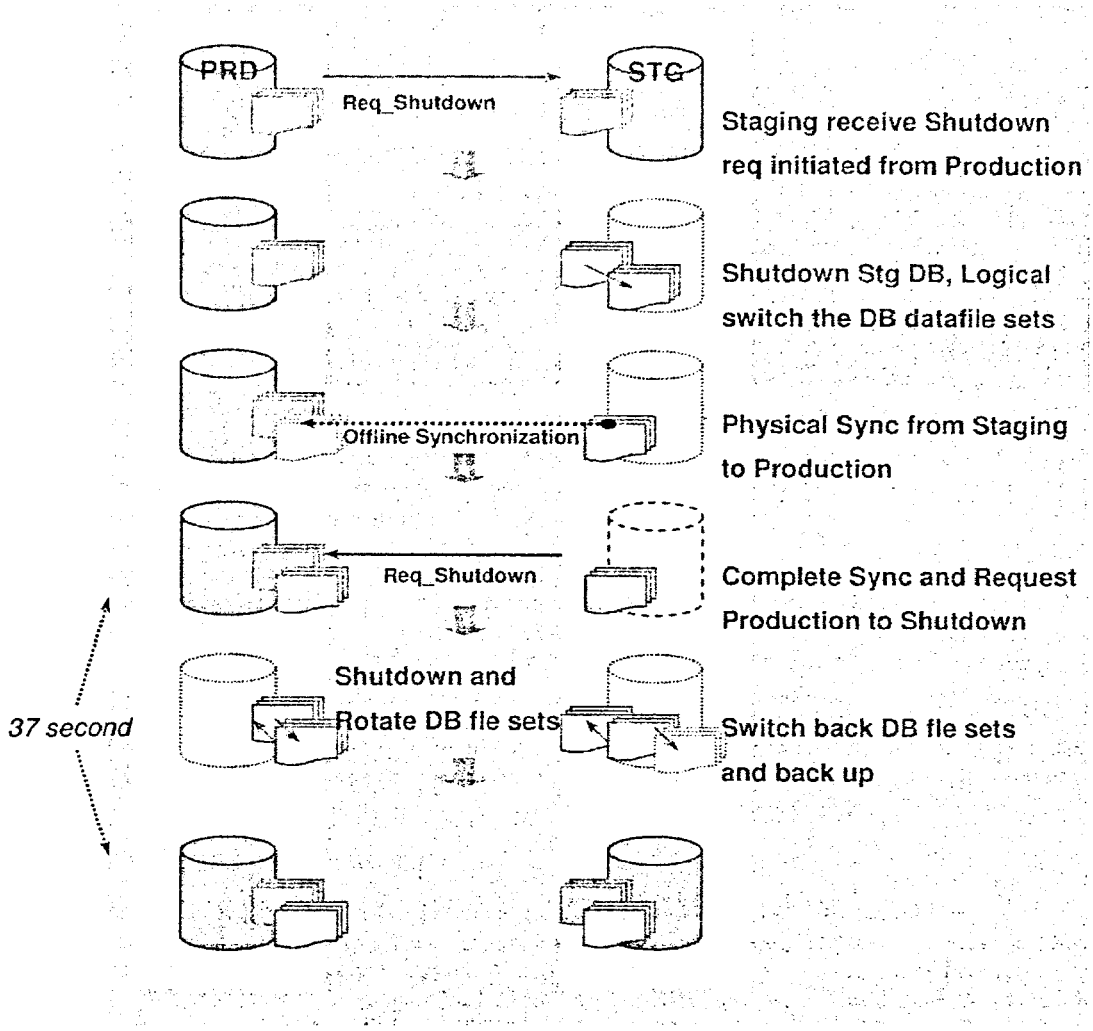
<u>Create Batch no.</u>
During the system validation process, system will generate a 10-dig batch no. for each batch in validati summary (date-time-series no.), that number will be created in the associ-ated priv table for each record and used for on-line process or future tracking

<u>Create new ID for insert case</u>:
ID= -1 means new-created record
In System Validation, system take the max+1 ID value from MRO, or NECA or ECALS depend on which group the table belongs

<u>Validation in Private table</u>

- <u>Before System Validation</u>: always Null
  Action
  I: Insert
  U: Update (after updated)
  D: Delete
  u: Update (before updated)
  null: rec never been touched
- <u>After System Validation</u>: (DB gen)
  S: successful, record is ready for staging
  D: duplicate record in the same day's upload
  E: record in stgDB exist for insert
  N: record in stgDB not exist for delete
  P: parent record not exist, cannot insert
  C: child record exist, cannot delete
  B: business rules violation
  F: format error、or action code violation
  R: range error

Model of Transition (Sync Production from Staging)



Restart Database of Production and then Staging DB, now ProdDB get 100% full image from StgDB, and both system complete backup

C9

Single View of Customer (SVOC)

IBM

For the presentation to ASEAN Enterprise Architecture

Version 1.0
October, 2006

Document Status
Version Number: 1.0
Date: 10/10/2006
Author: Raymond Wu, Consulting IT Architect
IBM Software Group

Summary

The new concept of SVOC (single view of customer) consists of two different areas, one is business innovation, the other area is IT information aggregation, the transformation of new concept in SVOC requires focus in the following dimensions:

- *Automates targeted marketing strategies across multiple channels*
- *Delivers catered, actionable information to channel employees at the point of customer contact*
- *Improves total relationship with the customer and increased sales*
- *Extends core banking applications to any employee device to increase productivity and accelerate sales*

Customer Centric Solution is based on Metadata technology and Concurrent run-time of ETL, EAI and EII to achieve Integrity, Intelligent, Integration and Performance, and play the leadership role in the industry

Customer Centric Concept

Customer Centric Solution is key to solve the common issues in Customer Data fragmentation.
Customer Data fragmentation means CIFs are fragmented, therefore no single customer's view and have difficulty to support a customer centric approach and to manage KYC, AML, Customer Profitability & Central Liability Management as consequential issues. The issue enforce the bank to use silo or part of the non-enterprise information, cause business high risk in data integrity, performance and future development.

- Help industry to provide a "one-stop" banking services, in retail, consumer and commercial segments, with customer information across multiple CIFs and operational systems
- Help enterprise to use a holistic view of it's customer's across all channels and business groups
- Emerging trends in new and complex banking products, investment instruments and services requires an integrated customer centric approach
- To administer and maintain multiple customer hierarchies (Risk, Marketing, and Legal). In order to leverage these hierarchies they require a holistic view of its customer to facilitate increasing regulatory requirements pertaining to KYC, AML and reporting
- The lack of a complete customer view, data stewardship and hierarchy management has directly impacted the quality of the customer relationship information available to the various channels and business groups. This could affect company's ability to derive some core customer metrics such as:
- Understanding the top customers that produce the highest net revenue for company
- Determining what customers / RM's have the largest increase in revenues Year over Year
- Identifying risk associated with a Customer based on approval of a proposed deal

As a result of these business drivers, company requires a customer hub to assist in the management and administration of customer duplicates and hierarchies. Some of the specific customer requirements are as follows:

- Solution must be independent of company's CRM and SFA environment, and provide services and UI capabilities to integrate with the current & future technologies

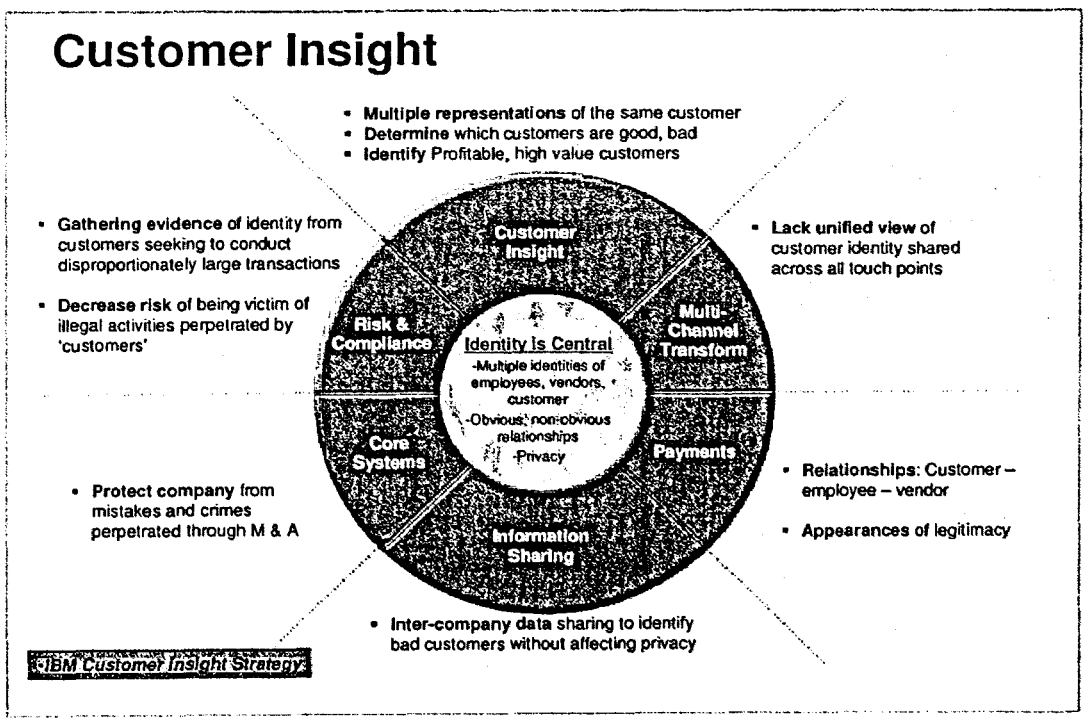
- Solution must fit within company's existing technology supported platforms and not require a proprietary environment (including application server, workflow tools..etc.
- Core Solution must be mature and have demonstrated implementations integrating across multiple environments

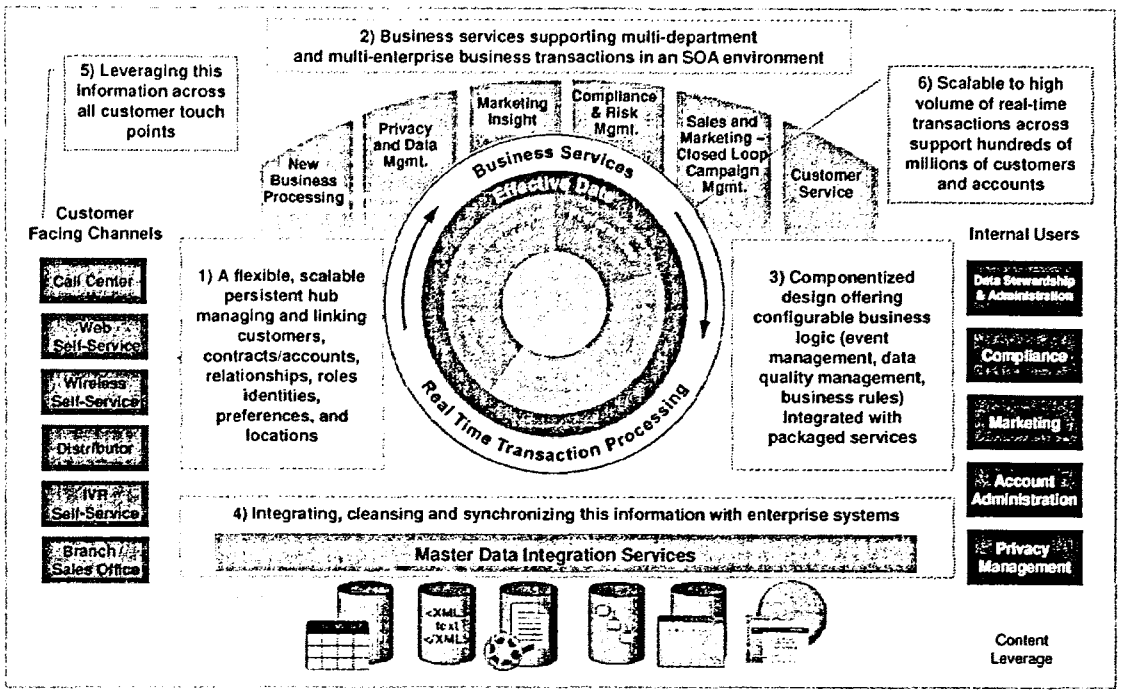
IBM propose the integration of its Websphere Customer Center as the customer hub within technology environment. IBM WCC is a real-time, transactional application which:

- Manages customer-centric business processes and contains enterprise business logic to maintain the single version of the customer truth including hierarchy structures
- Contains over 500+ business services and robust and flexible party-based data model ready for integration
- Contains internet-based user interfaces for administration of the IBM WCC solution, de-duplicate customer, manage hierarchy and display 360° view of customer
- Is fully componentized and exploits IBM WebSphere family open architecture (J2EE), enabling it to be extended and customized
- Is a proven solution for many banks and can perform and scale

Common issue

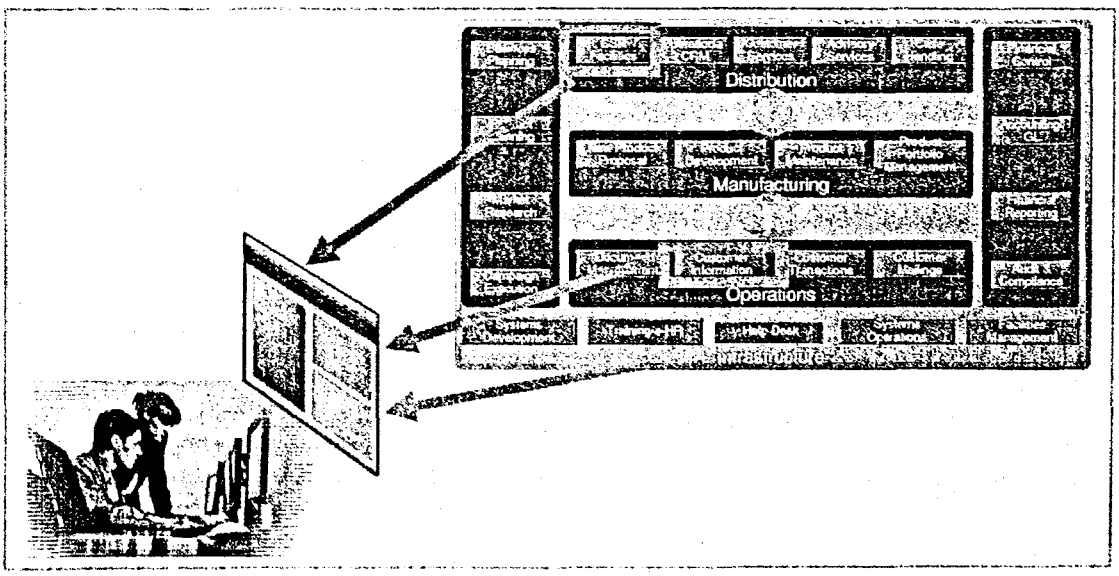
- Majority of commercial and retail customer data is distributed across multiple CIFs
- Difficult to adopt Customer Centric approach for other business functions including KYC, AML, Customer Profitability.
- Marketing and sales data in multiple sources
- No single way to access a comprehensive 360-degree view of customer data across the multiple LOBs within Lippo Bank
- IBM WCC to provide single point of integration of operational data to provide accurate and timely holistic 360-degree view of customer across the Enterprise
- Service oriented architecture to provide common access for all business units via service and/or message-based EAI infrastructure
- Neutral IBM WCC customer hub to integrate to current and future front-office and back-office applications and improve ability to integrate mergers and acquisitions
- Single customer data augmentation platform to improve knowledge and action on customer
- Flexible, easy to change, to improve time-to-market
- Scale, perform and support uninterrupted (7x24) online access

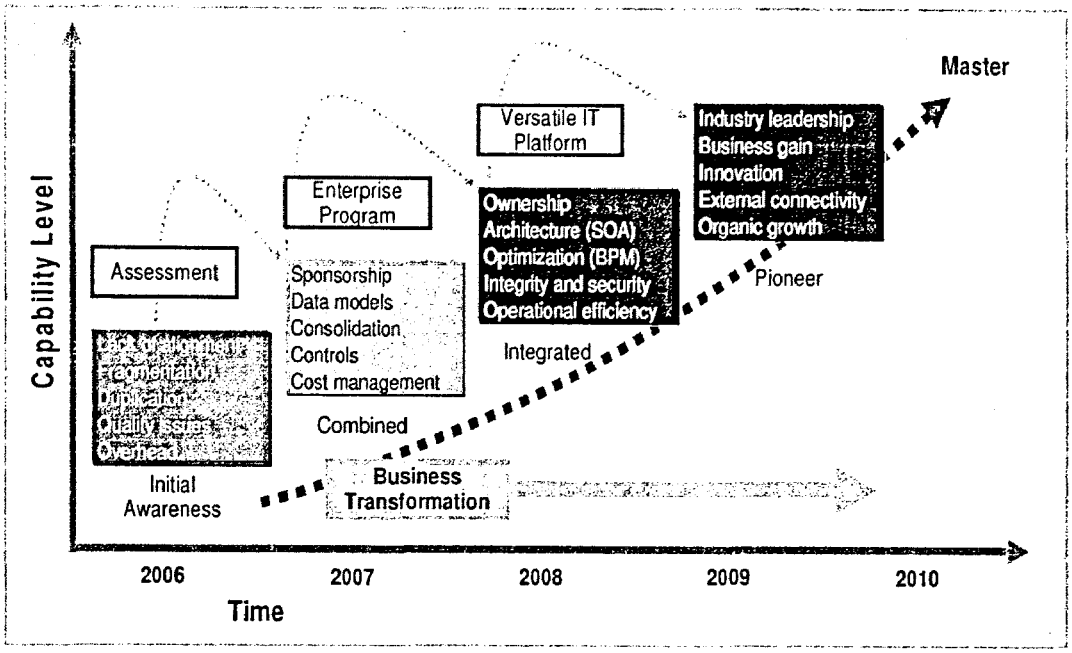




Portals Assemble Related Business Services

Each user (employee, customer, partner) interacts with a set of the business services to perform common transactions and collaborate with others to derive value from the infrastructure.





Information Aggregation

- Drivers
  - Unified view between multiple databases holding partitioned information that does not reflect the business model or desired services (e.g. due to merger / acquisition).
  - Tracking events or cases between multiple applications / agencies as they progress through a lifecycle.
- Database consolidation
  - Simplified data access services supported by single repository
  - May be difficult both technical and politically to migrate from distributed to centralised databases
- Data access virtualisation
  - Service aggregation and composition used to provide simplified data access services across multiple repositories
  - Care required in choosing roles of service integration vs. information integration technologies
- Data Warehouse / Operational Database / Read Mirrors
  - Use delayed feeds and a separate read-store if latency and duplication can be managed
  - Care required in choosing roles of service integration vs. information integration technologies

Model - Complete and proven DW design to provide complete integrated enterprise-wide infrastructure for all reporting requirements

Business Solution Templates (BST) - Set of reporting templates which cover all general reporting requirements of a financial institution. Each BST object is mapped to the BDW Model

Application Solution Templates (AST) - The data requirements list for non-reporting applications, such as risk weighted asset calculations. Current support for Basel II Credit Risk Engines

Project Views – A predefined statement of scope across all the models to solve a problem. Each view reflects the coverage and support for specific regulatory requirements.

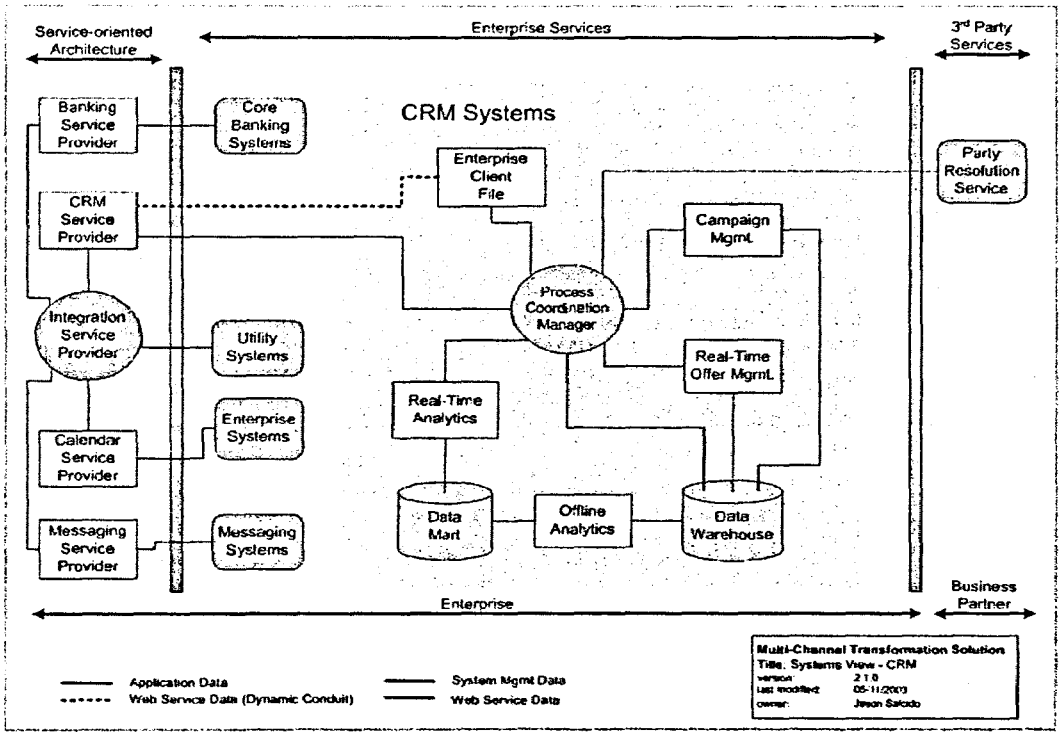
Mappings between the Models – Extensive set of predefined source to target mappings from the reporting and calculation data requirements list to the BDW Model to enable the rapid scoping of a design for a particular business issue

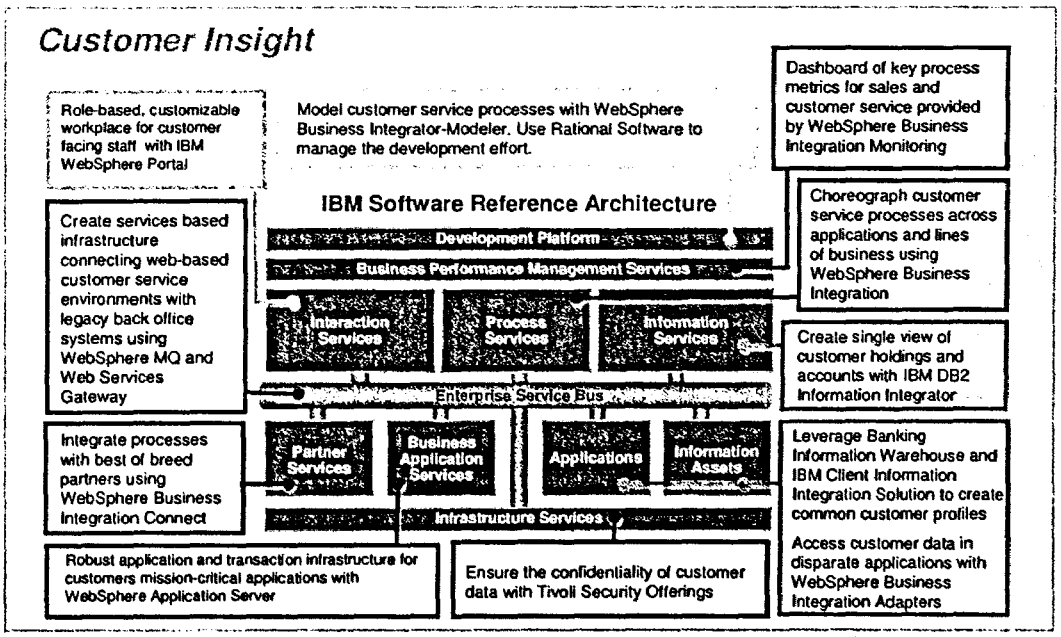
Business Benefits

- Improved sales and marketing metrics, with instantaneous feedback
- Ability to proactively contact clients at timely stages in their business relationship
- Ability to empower employees to service customers directly, without requesting approvals unless necessary
- Improved segmentation and deeper models, sends each customer three tailored offers monthly

2. Delivery speed for products and services increased through event triggers for next-business-day leads

Solution

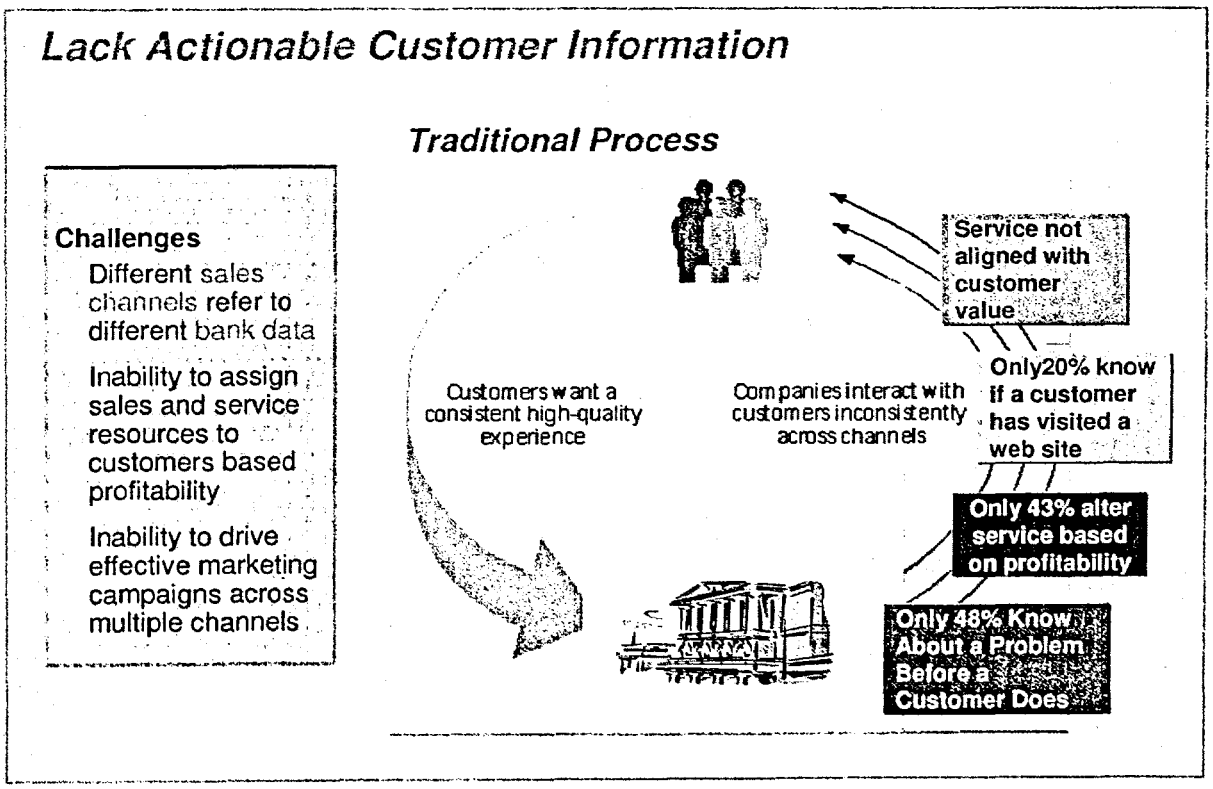


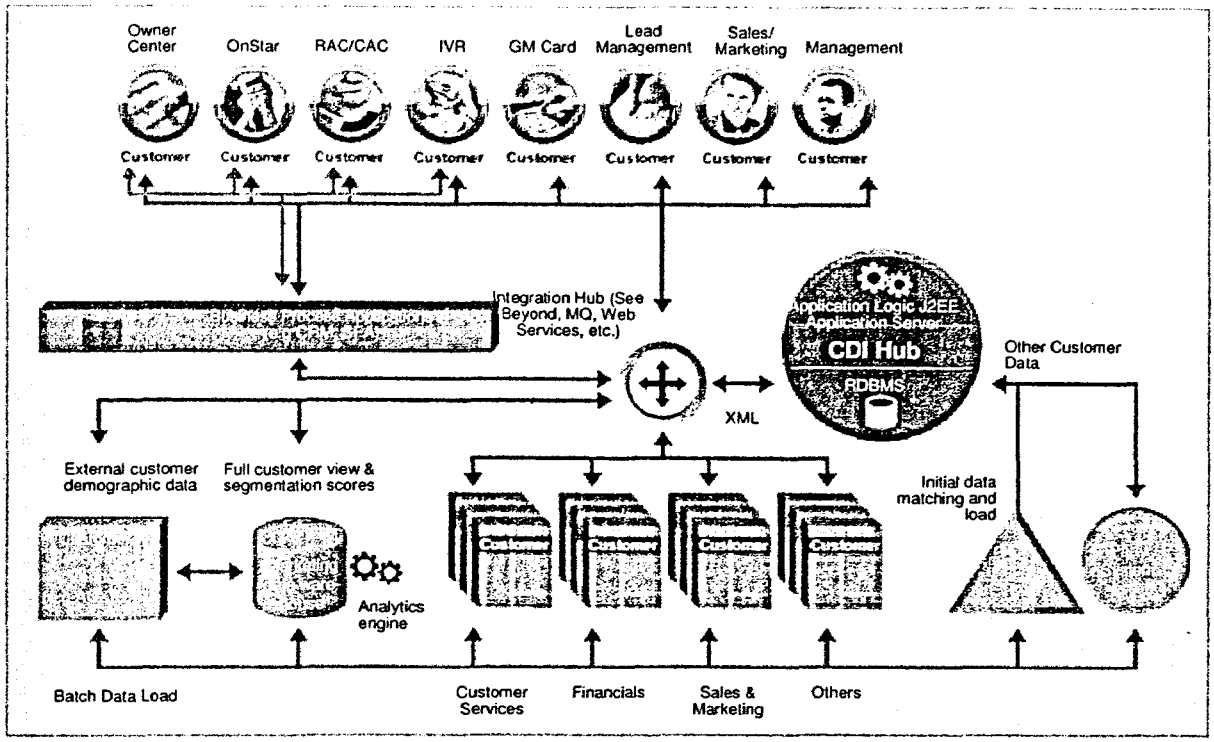


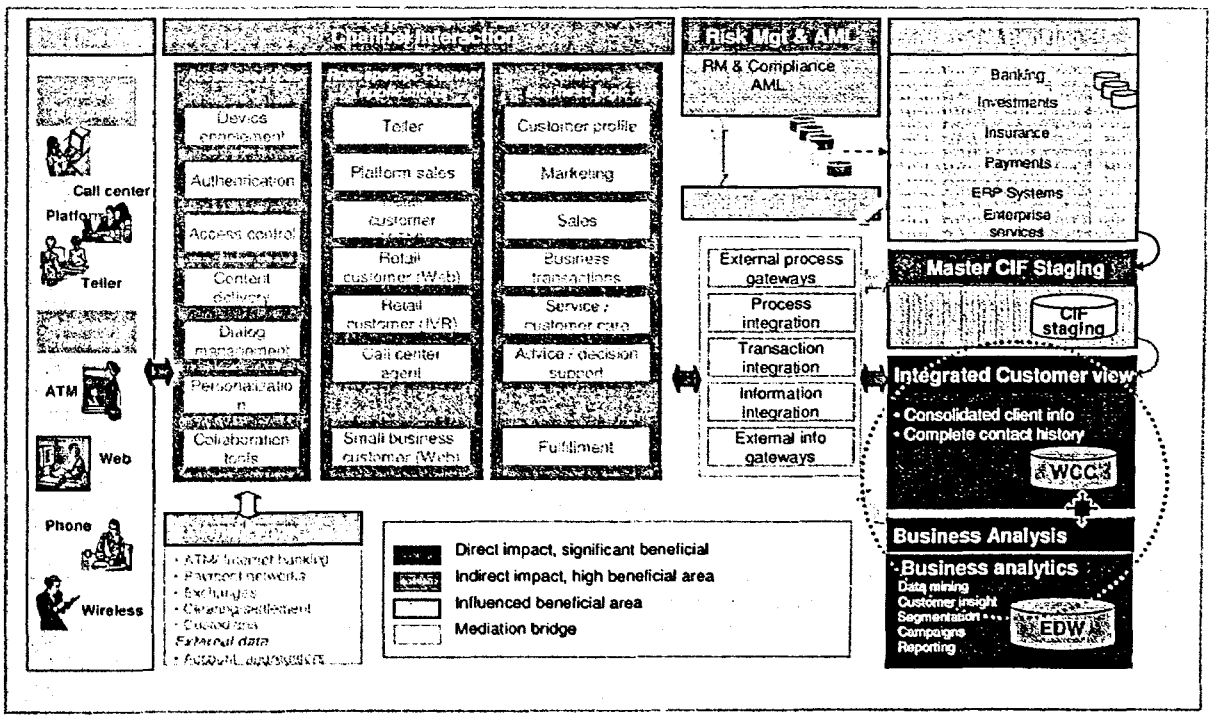
- A comprehensive customer insight solution from IBM, built on extensive information integration middleware, provides enhanced business intelligence, CRM, and scoring capabilities and Increased speed to making decisions

- BMO selected a comprehensive IBM Business Intelligence solution. IBM DB2 Universal Database (UDB) Enterprise - Extended Edition software running on IBM eServer pSeries 690 servers is the data warehousing complex for the solution. IBM DB2 Intelligent Miner Scoring tools provide customer segmentation, as well as business intelligence and customer insight services for the CRM analytical environment. With the centralized decision environment, all 7.5 million customers are scored monthly through DB2 Intelligent Miner

1	Build Enterprise Customer Information File	Enables a seamless view of client relationship with the bank.	Provides a complete customer view enabling cross-sell, retention, and credit management. Integrated enterprise-wide data model, information hub, and message hub solution. IBM Client Information Integration Solution and Banking Data Warehouse
2	Understand Customer Profitability	Understand Customer Profitability	Provides analysis and insight into customer profitability and product/service use for segmentation and differentiated customer service.
3	Increase Revenues through Campaign Management	Intelligent direction of Marketing Investments	Enables marketers to identify opportunities, perform customer selections, plan and manage campaigns, interact with customers and allow near real-time measurement and analysis
4	Analyze Markets and Customers for Product Development	Intelligent direction of investment in product/ service development.	Provides information for product/ service development and planning

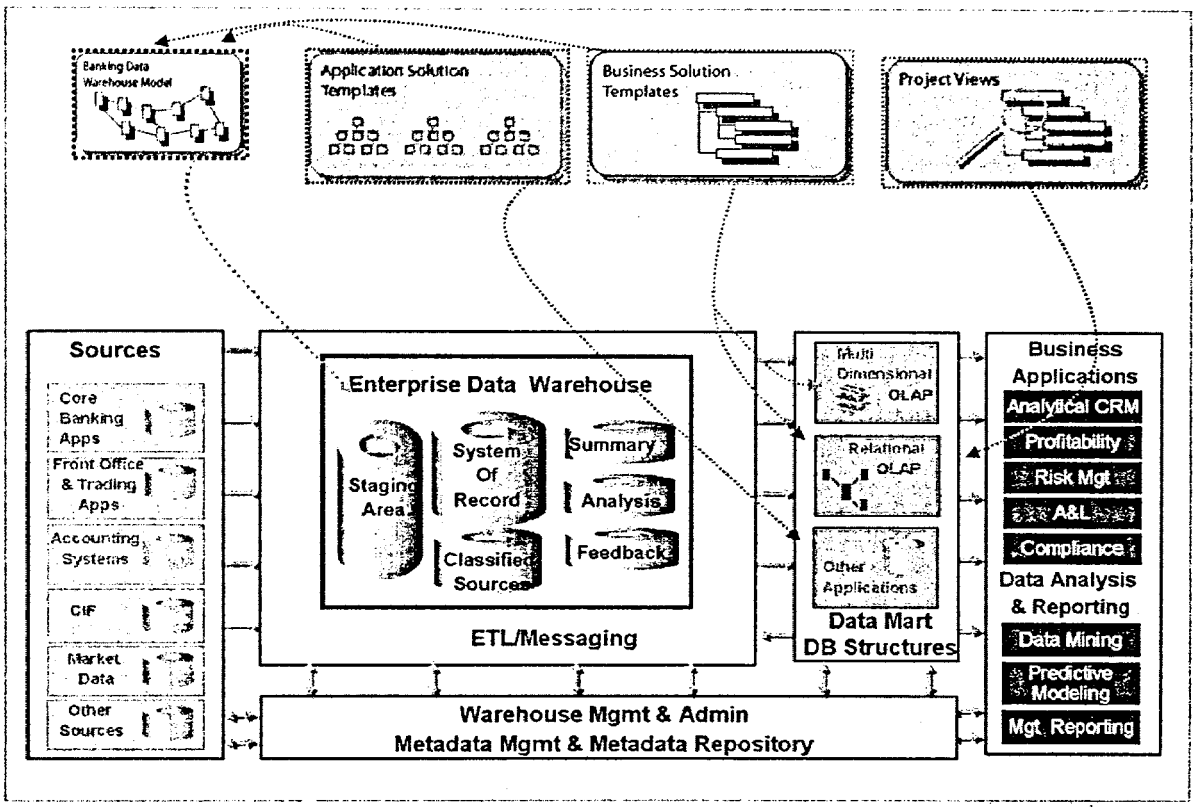



Owner Center
OnStar
RAC/CAC
IVR
GM Card
Lead Management
Sales/ Marketing
Management
Customer
Integration Hub (See Beyond, MQ, Web Services, etc.)
Application Logic J2EE Application Server
CDI Hub
RDBMS
XML
Other Customer Data
External customer demographic data
Full customer view & segmentation scores
Initial data matching and load
Analytics engine
Batch Data Load
Customer Services
Financials
Sales & Marketing
Others


Channel interaction
Risk Mgt / AML
RM & Compliance AML
Banking
Investments
Insurance
Payments
ERP Systems
Enterprise services
Master CIF Staging
CIF staging
Call center
Platform
Teller
ATM
Web
Phone
Wireless
Device management
Authentication
Access control
Content delivery
Dialog management
Personalisation
Collaboration tools
Teller
Platform sales
customer
Retail customer (Web)
Retail customer (IVR)
Call center agent
Small business customer (Web)
Customer profile
Marketing
Sales
Business transactions
Service / customer care
Advice / decision support
Fulfillment
External process gateways
Process integration
Transaction integration
Information integration
External info gateways
Integrated Customer view
• Consolidated client info
• Complete contact history
WCC
Business Analysis
Business analytics
Data mining
Customer insight
Segmentation
Campaigns
Reporting
EDW
External data
Direct impact, significant beneficial
Indirect impact, high beneficial area
Influenced beneficial area
Mediation bridge

Customer Centric Architecture

The customer-centric IBM Banking Reference architecture encompassing client's roadmap by taking CIF master data consolidation initiatives to drive single customer view

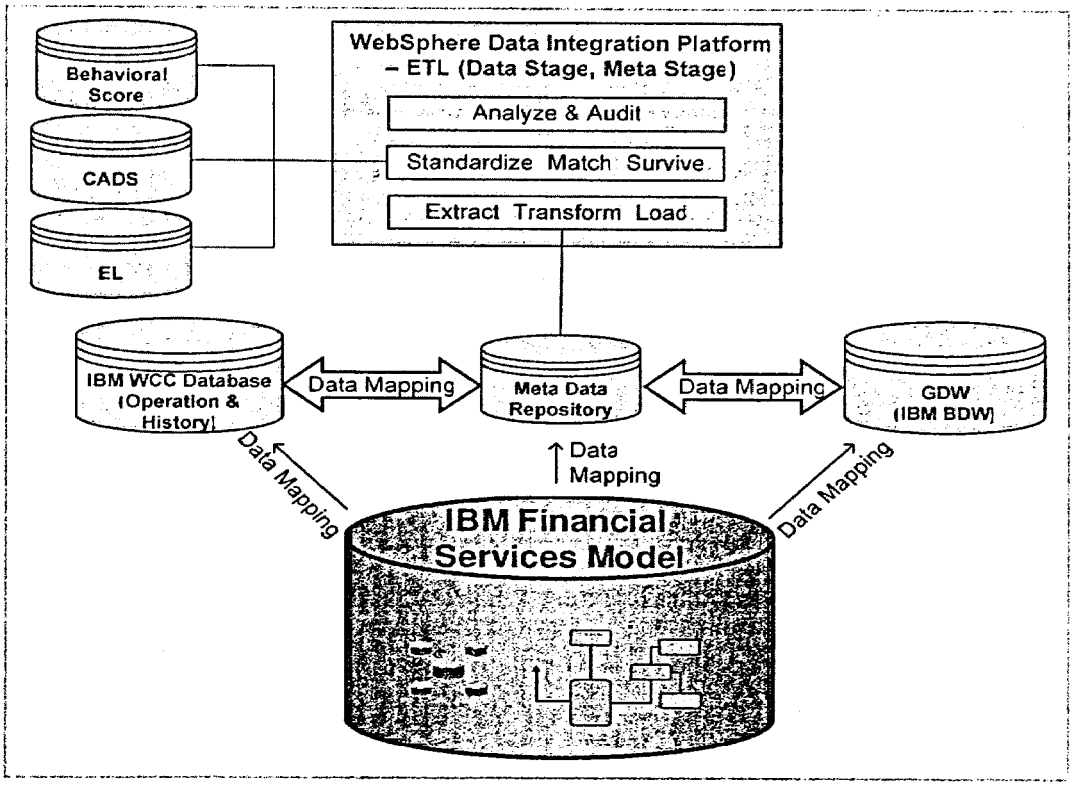


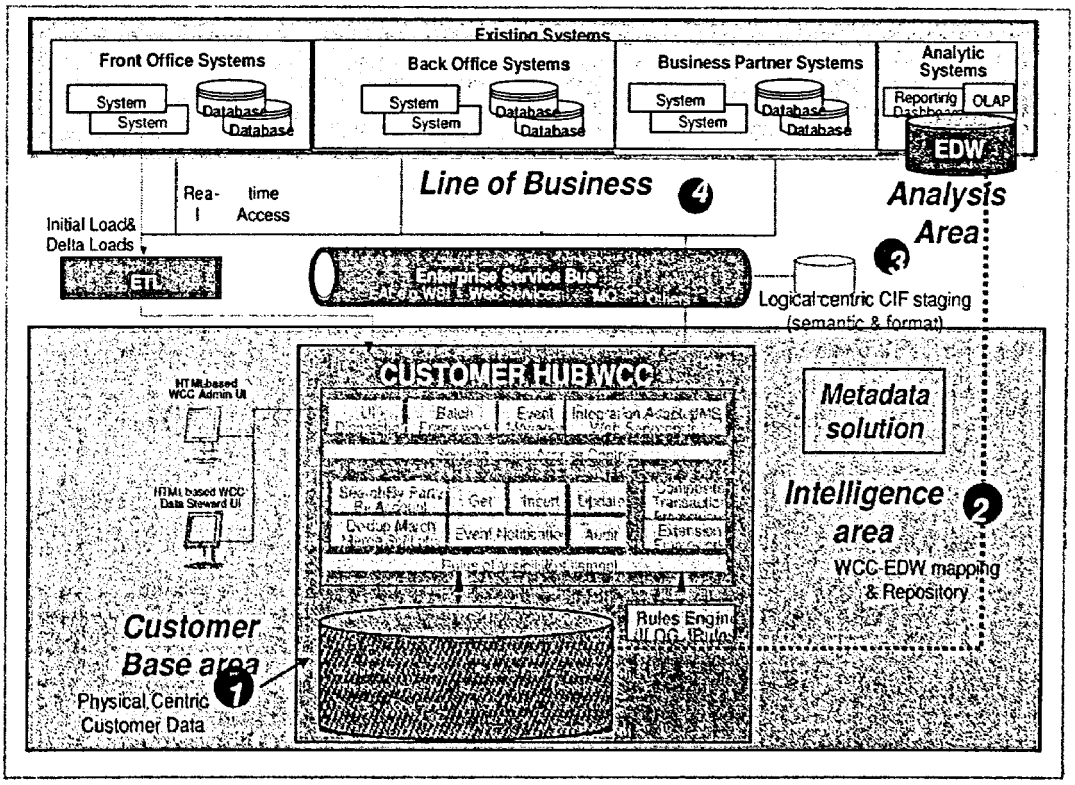
Metadata Integration

Cover industry requirements of real-time and batch job. Client gain advantages from concurrent transport of Websphere ESB and all EAI, ETL and EII combination of data integration solution.
The IBM Data Integration solution between WCC and EDW is by using metadata repository as physical solution and data mapping as logical dimension.

The Metadata repository holding meta information for mapping between WCC and EDW.
The Metadata repository synchronize with IBM Financial Services Model where holding the Standard Financial model and best practices as guidelines for Financial sector to follow.

There are 4 zones of Customer Centric Technical architecture – Customer knowledge base (centric CIF), Metadata Intelligence (most effective data integration), Analysis Insight (enterprise data warehousing), and existing Line of Business (front and back end). The coherence of IBM 4 zones solution provide best practice in Financial industry and optimized operational model.


Behavioral Score
CADS
EL
WebSphere Data Integration Platform – ETL (Data Stage, Meta Stage)
Analyze & Audit
Standardize Match Survive
Extract Transform Load
IBM WCC Database (Operation & History)
Data Mapping
Meta Data Repository
Data Mapping
GDW (IBM BDW)
Data Mapping
Data Mapping
Data Mapping
IBM Financial Services Model


Existing Systems
Front Office Systems
Back Office Systems
Business Partner Systems
Analytic Systems
System
Database
OLAP
EDW
Line of Business
Analysis Area
Rea- l time Access
Initial Load& Delta Loads
ETL
Enterprise Service Bus
Logical centric CIF staging (semantic & format)
CUSTOMER HUB WCC
Metadata solution
Intelligence area
WCC-EDW mapping & Repository
Customer Base area
Physical Centric Customer Data

C10

Architecture Rationalization Part II - Corporative Strategy



Version 1.3
January 7, 2006

Prepared by: Raymond Wu
Chief Technology Officer
EDS Greater China

Document Status	Regional compliance
Version Number:	1.1
Date:	01/10/2006
Status:	Reviewed

Document Revision History

Version	Authorised Date	Reason for Change	Author Name	Responsibility
0.1	December 15, 2005	Basic information	Raymond Wu	Chief Technology Officer
1.0	December 30, 2005	Initial draft	Raymond Wu	
1.1	January 10, 2006	Reviewed	Raymond Wu	

EDS' qualifications, without the prior written consent of EDS.
EDS is a registered mark of Electronic Data Systems Corporation. EDS is an equal opportunity employer, m/f/v/d.
Copyright © Electronic Data Systems Corporation. All rights reserved

Summary

Rationalization is required when we work on those root issue of common IT pains, following paragraph gives examples of typical IT pains and their root issues, rationalization should be started from development environment instead of production, as a good design need a good development environment

The common IT pain is not limited to the following issues, they are the usual case that need our attention in our development environment, as we believe a good design normally come from a highly rationalized environment. *The research of architecture rationalization is classified into the following areas –*

- *Infrastructure*
  Infrastructure need a review and ensure all the necessary version upgrade (OS, Platform, DB) has been done. *(level 1 - short term plan)*
  In the future plan, Financial industry need a focus in legacy modernization for multi-tiers architecture, while Microsoft will continue to be the target for Government industry. *(level 3 – long term plan)*
  Governance and enterprise level control is still the key to prevent redundancy and fragmentation. *(level 4 – concept only)*

- *Tools*
  Development env. need robust, CM and testing tool *(level 1 – medium term plan)*
  Borland StarTeam, CA Endevor and Mercuri are all assets and EDS standard tools ready for deployment. *(level 2 – medium term plan)*
  Programming standard is reviewed for code compliance and QA. *(level 1 – short term plan)*

- *Project*
  Roles to support project SDLC - SME, Architect and Technical management. Enterprise project architecture is in demand in GC&K for high complexity mega deals. Rationalization is critical for technical leaders to commit the healthy function of these 3 roles *(level 2 – medium term plan)*

- *Client*
  Common interface and business model are needed, to clearly define and rationalize the boundary between client and EDS. *(level 3 – long term plan)*
  Common interface minimize the risk and protect EDS's responsibility from running across the interface into client's site. *(level 3 – long term plan)*

- *Applications*
  Pre-build common component is needed to strengthen the architecture foundation, minimizing the gap and agile in each development. 70% foundation (from 30-50%) is targeted, it should be the focus of development environment rationalization for 2006. *(either level 3 – long term plan or level 4 – concept only)*

- *SLA*
  SLA is related to the stability of Infrastructure and Application, there is no direct impact of development infrastructure to SLA.

3. Corporative strategy in rationalization

3.1 Strategic rationalization in eliminating dependency case study - Workflow Java engine to replace Lotus Notes

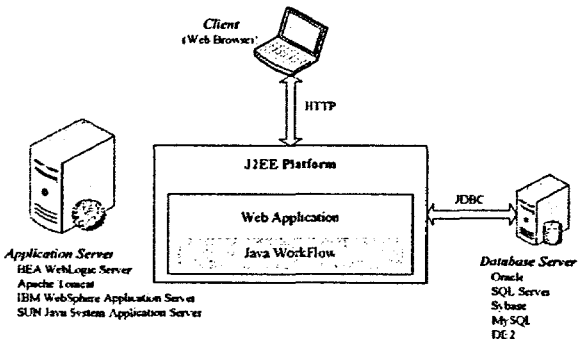
- ✧ We try to avoid the dependency of solution provided by our competitor
- ✧ Lotus Notes is weak in computing and reporting, Non value-added, Poor integration capability, security, and License Fee is involved

- ✧ Approval process module is the core design in EDS Work Flow Java version. In each approval sheet he parameters are fully applied and the Framework allow designer to configure their new sheet in a fast way and customization capability, any document only need the basic approval information and can bee added to the process easily

3.2 Multi-environment application rationalization

1. Consolidation
   Whenever feasible, consolidating before integrating system, so the first choice should be consolidate disparate systems that perform similar functions into a single system

2. Integration:
   Integration dimensions include data, process, platform, network, organization, location, employees, customers, product and data-centric approach to integration has been one of the most prominent dimension



3. Event driven model:
   Business event model is the common integration element which drive all others and the container that bring the process and data dimension together, all the end-to-end business process driven through business event model across to all other dimensions

4. Enterprise and Unique:
   Enterprise need a multi-tiered, non-redundant and coarse-grained architecture so end-to-end integration solution is provided by distinct layers and each of which provides unique and non-overlapping services.
   Therefore architecture must emphasize reusable component building block to maximize the enterprise potential for rapid change, agility and flexibility, Service oriented architecture and Commodity architecture should be adopted which will take advantage of standard, readily available common

3.3 Project rationalization - Governance of project architecture in high complexity development

In the following example, team leaders (Data, System, BA, SA, Dev, Conversion, Testing) form a virtual team and communication in project governance, the mechanism of the process route is to predict the defect in a collaboration design.

The strategy of this project architecture governance is try to create the route and communication channel for technical leader (from BA to development) to log their concern and change request, so all the system and data experts can work on the business driven issue and provide system level provisioning to support the application, to form a healthy information architecture.

3.4 Re-engineer the well-defined interface between EDS and client
Once we have a well-defined common interface for a common architecture, we can eliminate those ambiguous scope and client site transformation by using this common interface, so client is responsible for taking their existing business component, data, metadata and process functions and convert into this common interface, EDS's responsibility is transform from common interface to EDS's architecture.

This minimize the risk and effort for both EDS and client, what we need to do is minimize the scope need customization to be less than 30% (ideally 0%).

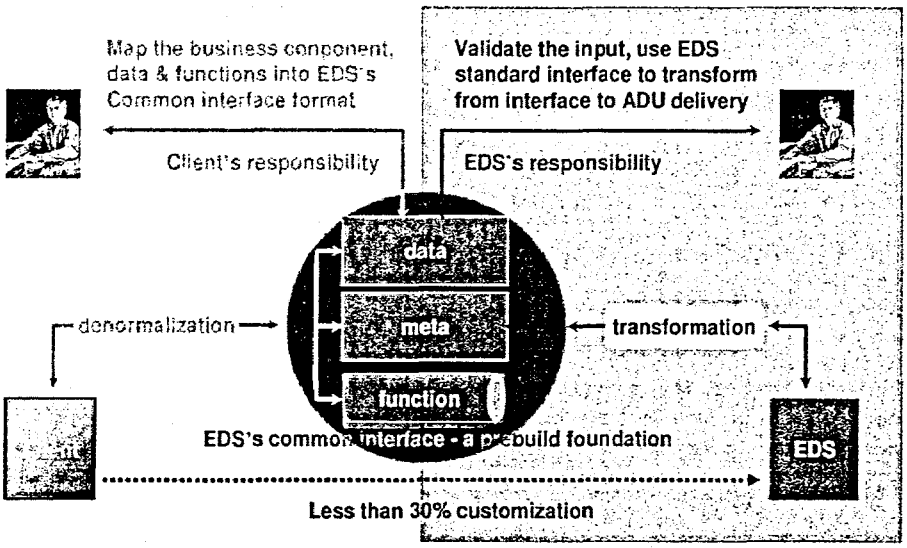
The first step of this concept is creating a common interface for generic purpose. Working on the format and schema of this interface need a survey of industry common requirements and EDS's existing architecture, once this is done, we need to work on the process that will take the interface to existing architecture.

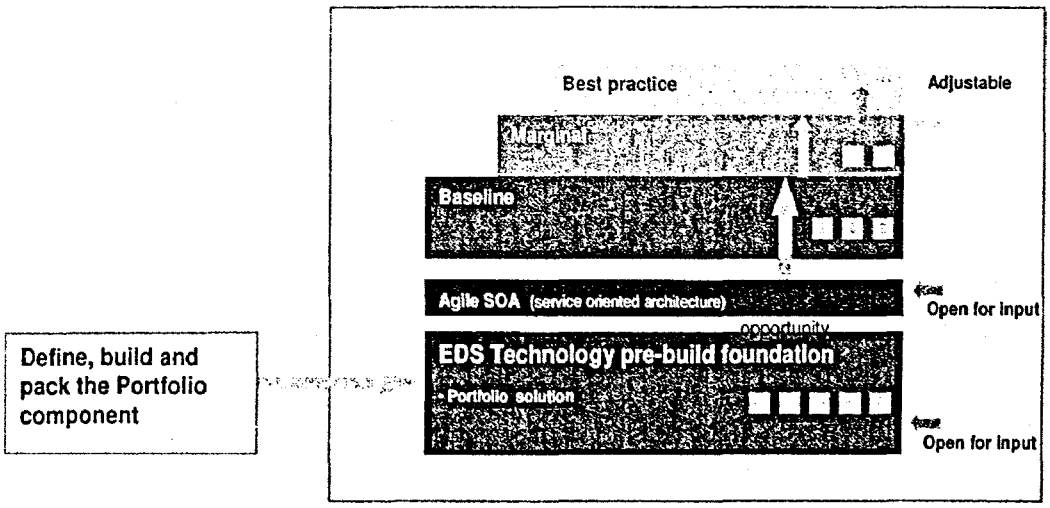
3.5 Foundation rationalization -
Re-engineer EDS architecture foundation

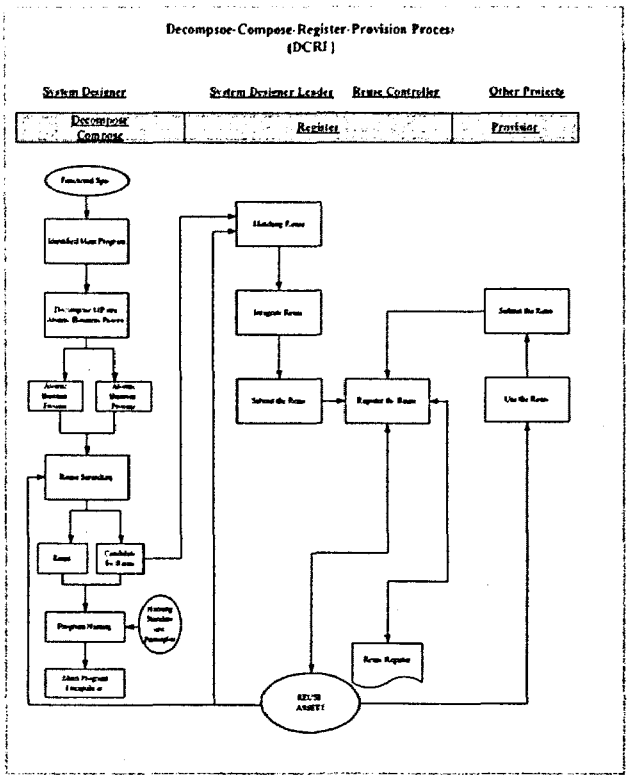
1. Use existing architecture components, EDS portfolio, methodology, business element and external commercial (common component) to build up the coarse-grained technology foundation as the candidacy of baseline foundation

2. SOA can be applied to orchestrate and reuse such component to fulfill the services

3. Most of the effort should be spent on the gap between "baseline" and "marginal", between marginal to "best practice" is adjustable based on the optimization from case to case

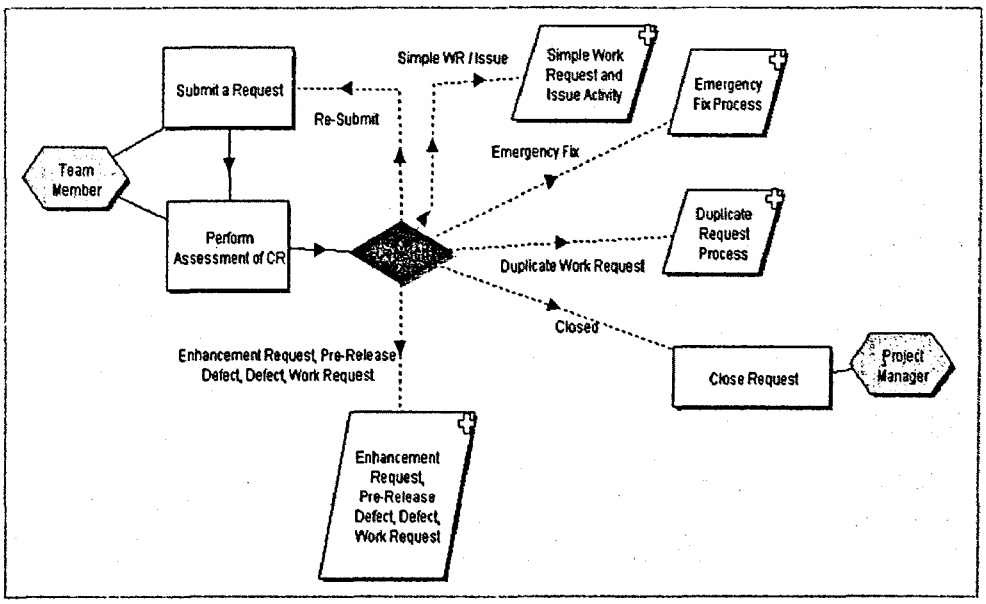


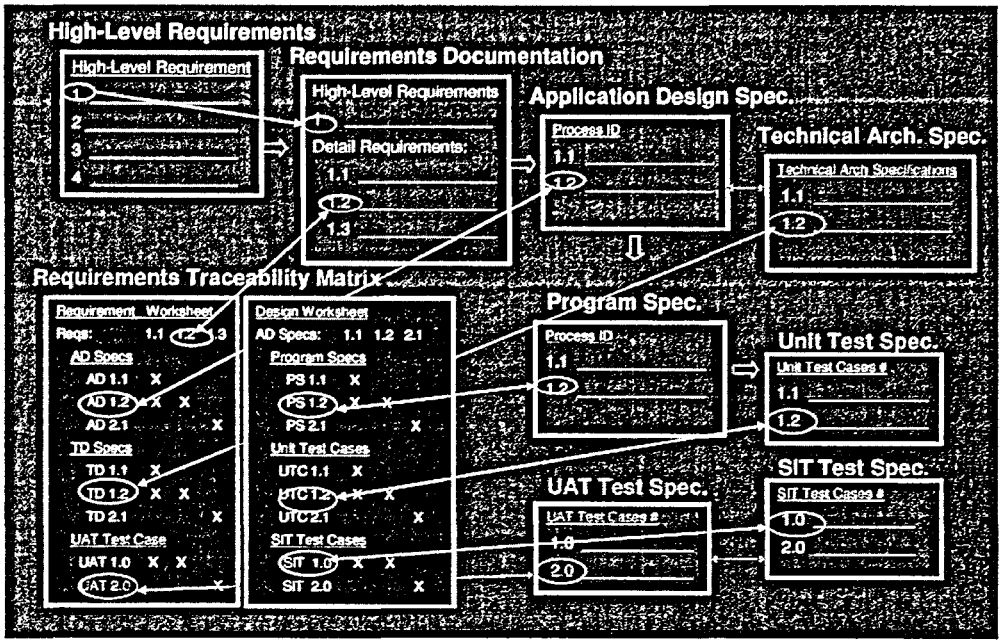




Test key items – what need be tested

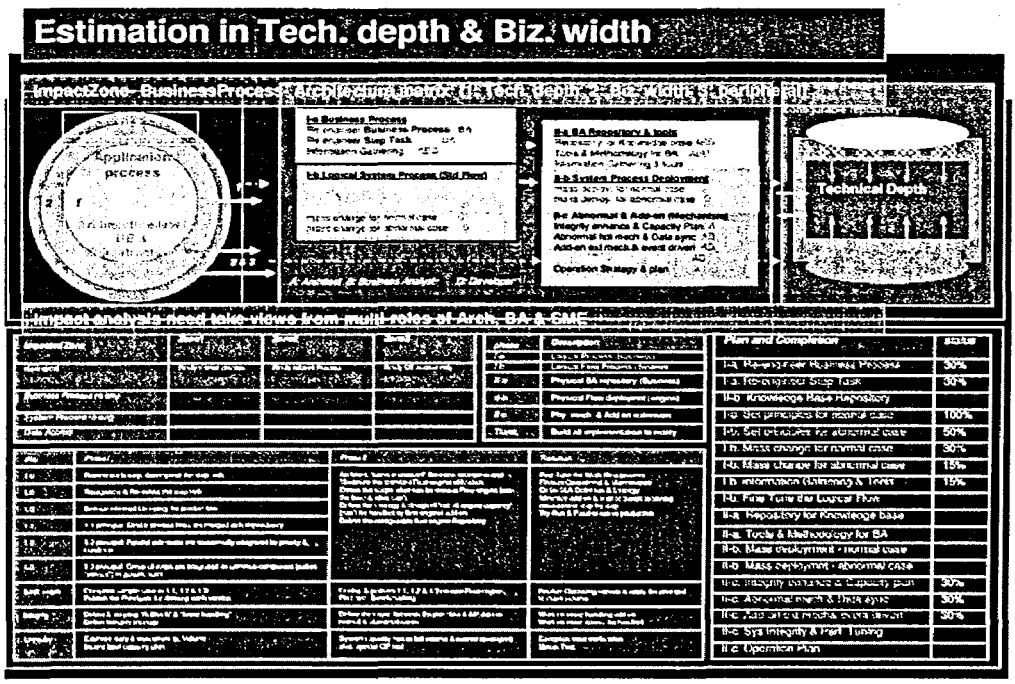
When evaluating a change request, the impact to all work products needs to be considered. The requirements traceability section is used to identify the work products that will be affected from each phase of development. This ensures that the full impact of a change is understood. When evaluating a change request, the requirements traceability information can be used to help determine the work products impacted by the proposed change.





Estimate key items – what need be estimated

When analyzing the impact caused by a change request, the impact to all dimensions needs to be considered. This means all the business scope, architecture technical depth need be evaluated, impact zone need be classified into 3 (direct core impact, partial impact and peripheral impact), each zone need be assessed in estimation process, this ensures that the full impact of a change is evaluated. Therefore the scoping and estimation will be multi-dimensions instead of simply from business scope.



Strategy



S1 Validation and analysis process in Component Based Design

IEEE Southeast

S2 Enterprise Integration Framework and metadata

DIIDS

S3 Enterprise Integration in Metadata Environment

IOL

S4 Case study of micro component design and industry integration

DIIM

S5 Enterprise Interoperability and Service Virtualization

ACM NZ

S6 DGRP Strategy in SOA Realization

IBM Corporation

S7 Impact Analysis of Platform Selection

Vodafone Japan

S8 High Performance System Design

IBM Corporation

S9 Telecom Architecture and eTOM

Oracle Corporation

S10 Telecom Billing and Rating Architecture

Vodafone Japan

S11 Architecture Recommendation in Public Sector

IBM Corporation

S1: Validation and analysis process in Component Based Design

Publication: Southeast07 (IEEE) Conference Procedeing
22 -25 March 2007, Richmond, VA, USA (http://www.southeastcon.org/2007)
ISBN: 1-4244-1029-0

Publication Contact: Professor Gang Tao
Technical Conference Chair
Dept. of E.E., University of Virginia
gt9s@ece.Virginia.edu

IEEE

IEEE SoutheastCon 2007

Date: Tue, 23 Jan 2007 13:19:40 -0800

From: gt9s@ece.Virginia.EDU Add to Address Book Add Mobile Alert

To: rwu_academic@yahoo.com

CC: gt9s@ece.Virginia.EDU

Subject: Your SoutheastCon 2007 Submission (Number 97)

Dear Mr. Raymond Wu:

On behalf of the SoutheastCon 2007 Technical Program Committee, I am pleased to inform you that the following submission has been accepted to the 2007 IEEE Southeast Conference:

The Validation Process and Component Analysis of Enterprise Integration

as a full paper.

Sincerely,

Gang Tao
Technical Conference Co-Chair
IEEE SoutheastCon 2007

Reviewer's Scores

Originality: 4
Technical strength: 4
Presentation: 3

Comments

This paper gives a new outlook to organizational structure and hierarchy for any type of business or project. The paper explains how the hierarchy and model of an organization can be set up, the factors that need to be taken into consideration and importance of each layer in the hierarchy starting from the Stakeholders and decision makers to the actual implementation.

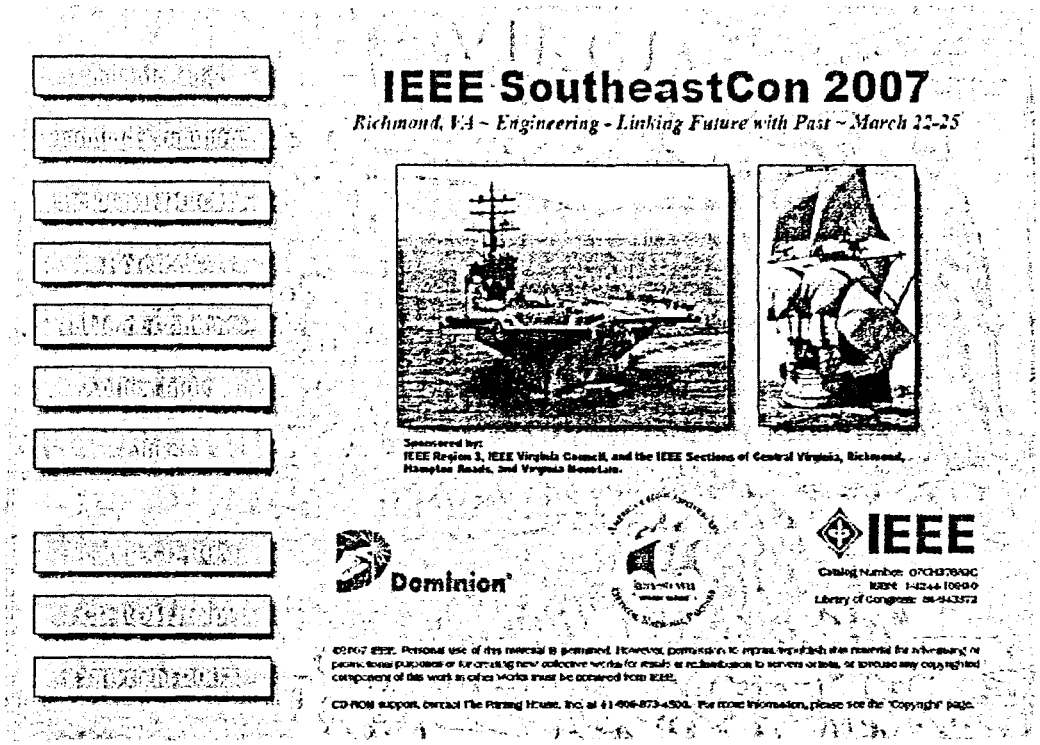


TABLE OF CONTENTS

Scroll to the title and select a Blue link to open a paper. After viewing the paper, use the bookmark "Go to Previous Document" to return to the same page in the Table of Contents.

Computing and Real-Time Systems

Validation and Analysis Process in Component based Design 208
Raymond Wu

Web Process Design and Services Virtualization 214

Designing a pH Data Acquisition and Logging Device using an Inexpensive Microcontroller 217

Electric Utility Migration to Regional Transmission Operations: A Case Study 221

A New Frontier for Real-Time Systems: Lessons from Molecular Biology 227

Software Safety Risk in Legacy Safety-Critical Computer Systems 233

Semantic Metadata in Enterprise Integration 239

An Embedded Linux based Navigation System for an Autonomous Underwater Vehicle 245

Case Study of Component base Integration in the Financial Industry 251

Validation and Analysis Process in Component Based Design

Raymond Wu
IBM Software Group
raymond@de.ibm.com

Abstract

Decision input from human factors play crucial role in component-based design. In the past decade, much of the alternative decision in criteria and preference evaluation have been massively implemented by group decision. In this research, we extended the focus of the methods and techniques to component decision in supporting the enterprise integration and ontology interoperability, we believe, to follow the nature of business in performing human interactive in this early analysis and pre-design stage, play a key role in the integration's success, we have assessed the feasibility and received the consensus of pre-design stage of industry.

Introduction

We continue the previous strategic work in enterprise integration, with further investigation of the human factors in interoperability within enterprise integration. Vertical enterprise integration is a metadata driven approach in the component-based environment, the first four from the total ten steps within our enterprise integration approaches are decision process related, they are:

I. Validation process of business components
II. Validation process of service components
III. Validation process of physical components

We start from the decision-making process of business component selection to derive the service components and physical elements. The decision-making process uses group decision-making method to take each individual opinion or preference and multiply it [1]. The decision makers are motivated to communicate and share opinions within the group in making business component selection. We use the same political model in all business, service and physical component validation, based expertise. The reasons are we use framework in keeping alignment between these different components in building up the decision and significance assessment of their interoperability instead of merely a particular component as a result.

Both political and rational models which we will introduce in a separate research, are constructed in a component-based environment; this is because in a scrambled development environment provides no opportunity for group decision-making or rational analysis. With all the contributing factors in a mixed code, any changes or add-ons can cause a huge impact. Even the scope of the impact area and components can be difficult to determine, thus the impact is high while the interoperability is low. In conventional development, weak visibility between business, service and information makes the linkage and enterprise interoperability impossible.

Once the alignment is missing or invisible between layers, the common services and service virtualization become impossible as these require a foundation in both macro interoperability and micro mechanism. This results in a silo system with serious limitation of services and weakness in enterprise integration [2], as a matter of fact, enterprise integration gain no benefit from group decision and deviate from the nature of business. As mentioned above, the methodology we used in this research work included the component model, political model, and macro metadata integration; we differentiate these with conventional methods in the criteria listed in Table 1. In the following paragraph, we demonstrate how to use the identified steps to derive the coherence of their interference in building up enterprise integration. Before we proceed to the key area of these steps.

We use architecture terminology "K-B-S-C transformation" to represent interoperability between architecture layers. We began with enterprise strategy (K), the chief area of concern for stakeholders and business managers, to derive the components in both business (B) and services (S) layers. As each layer of K (context), B (conceptual) and S (logical) are all at the conceptual or logical level, the bottom layer which is the physical component (C) forms a loosely-coupled alignment with S in provisioning bi-directional interoperability. The micro mechanism between these two layers forms the service unit; aggregate individual service

1-4244-1029-0/07/$25.00 ©2007 IEEE.

provides service in the business layer to finally achieve the strategic goal:

K: KPI - strategic Key Performance Identifier and all concerns of the stakeholder

B: Business Component - each block perform independent business

S: Service element - logical service component in atomic level

C: Component element - Physical component associated with S and B

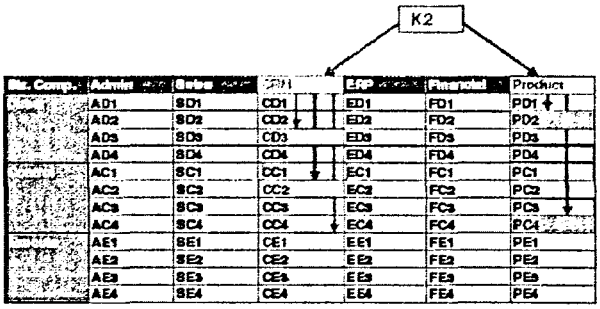


Figure 1. Business component active map produced by group decision

Stage	Conventional	Component based
Analysis	On-demand full domain	Subset of domain analysis
Architect	Use-case driven through full domain	Component based driven and
Development	End to end development from biz to data	Component selection in a small scope
Integration	Gap exist from domain, timing and project	Alignment in both vertical and horizontal are clear
Development effort	High effort in new dev. No reuse is available	High effort in new dev. Reuse is available
Risk	High, as scoping is difficult	Low as scope is clear
Maintenance effort	High, any change cause big effort	Low as each change can be identified and reused
Service level	Low to medium with constraints	High, service virtualization

Table 1. Benchmarking of integration methods in conventional and component base

Literature Review

S. Melnik (1999) proposed a universal interface which avoids common models and languages, thus the generic interoperability framework was developed to facilitate the integration of heterogeneous information systems. Within this framework individual communication protocols, data manipulation languages and data are represented in a generic manner preserving their ontological variety.

This makes it possible to build a mediation infrastructure using canonical wrappers, operational requirements of integration and layered architecture for component design [3]. Based on this common framework, we have the opportunity to build the component of both functional and non-functional elements utilizing the industry standard. The standard or common applications of the component we use in provisioning the interoperability, we called them common component.

DSouza specified the standard and common component as follows: "To manage models of large scale, a coherent modeling architecture is required, with standard ways to describe shared concepts, rules, patterns, frameworks, mappings, and generators." For example, the development of different applications with object models mapped to relational tables requires a single object-to-relational mapping architectural style from a shared model as a refinement framework. If all the enterprise models maintain a separation of information, engineering, and computational views which are then integrated with them in particular ways, this common structure should itself be modeled in a shared architectural style model as a package pattern [4].

Lagoze and Hunter (2001) build a conceptual model to facilitate interoperability between metadata ontologies from different domains. Their model uses Entity as the root class and assigns three categories Temporality, Actuality, and Abstraction as its subclasses.

This imply the communication between domain ontology is possible, however the business decomposition was not covered in the literature. In our ontology approach, business component grouping and validation are all the strategy and initial steps in establish the component based service provision in vertical direction, Figure 1. shows business decomposition.

The main purpose we apply group decision in component validation was to assess and identify the impact areas starting from business components in deriving the associated service and component for further implementation. This has been the main contribution in group decision literature as this has been seldom applied in component based development, furthermore, we took the result from component validation in common service analysis which is the small subset of Ontology and subset of business components which we called active map, together with aligned service and component for further analysis to determine the common service and aggregation.

Component Based Decision Process

We will now extend our focus to each step and demonstrate the decision and validation processes through these steps.

I. Validate business components through group decision-making

The initiative of enterprise integration strategy has three key tasks to fulfill: service validation, service transformation and service provisioning. Steps 1, 2 and 3 relate to service validation, steps 4 and 5 to service transformation and step 6 to service provisioning. Validation is mainly based on the assessment of business criteria via human decision-making with the scope of validation covering each layer of B (business), S (service) and C (component element). Once business components are selected for a project with alignment to K, the derived services from B will be assessed and validated. The semantic metadata within K-B-S-C therefore provides a means to hold their matrix alignment; for design purposes, the referential integrity and semantic interoperability require that there be a repository to hold the information. The K-B-S-C metadata integration forms a skeleton in enterprise integration with the choreographic capability of retrieving each layer's metadata for the details of each component stored in the metadata repository of B, S and C. This macro to micro hierarchical structure constructs a vertical interoperability skeleton in which the different roles work together; the architect takes care of the main route of the skeleton in metadata integration and the developer takes over the skeleton's work and extends the micro design in the transformation of components and intercommectivity of service.

Service provisioning is based on an assessment of frequency of use of a component and its dependency within a project. Before implementation, the strategic KPI will first be reviewed based on a project's requirements followed by business component selection and validation. Once the business components have been validated, the deconstructed service components and associated component elements will be evaluated and validated, using the same validation methodology [5].

As each project needs such an assessment to determine the composition of K-B-S-C and the significance of each component regarding its frequency and dependency in provisioning validation, there are often judgments to be made. These involve both human evaluation and use case parsing to determine the components to be used and priority of deployment. The goal of a project may be a business motivation or an IT related task; in either case an enterprise vision is required to maintain the strategic enterprise coherence of each project activity across variously compliant organizations. As mentioned, the first step is a "group decision in producing an active map" meaning that a group discussion is required for expression of opinions. A typical group decision-making team consists of the board of the company, business manager, enterprise architect, and program manager.

There are two factors directly influencing the formation of an optimal group decision, the first being the role of the individual. Although each decision maker tries to influence other members to adopt his or her viewpoint, powerful members will sway the decision-making more strongly than will other members. Group members thus have different "weights" in a group decision-making process; this fact should be taken into account in the generation process of the satisfactory group decision. The second factor is the individual's preference for specific components. The final decision on a business component and its associated service elements needs vision from corporate strategy, business goal and enterprise architecture in order to ensure that all the different viewpoints shed light on a common focus. This is true not only in the selection of all the components necessary for the project but also to create visibility of criticality,

1-4244-1029-0/07/$25.00 ©2007 IEEE.

frequency and dependency of components within the project.

Some projects may present a clear picture of fixed solutions in a given criteria set; however, many projects have alternatives which need to be assessed. For example, if the goal is to increase revenue by 30% in the year 2007, potential decisions on business components may vary widely according to individual perception and focus, creating diversity in opinion and randomly distributed components. Indeed, each opinion can be valuable as a political model in a component-based architecture helpful to business innovation.

Layer	Typical role	K-B	B-S	S-C	S-S
Strategy	Stack holder	H	L	N	-
Business	Business Mgr	H	M	N	-
Service	Architect	L	M	L	-
Component	Developer	N	L	H	-
Enterprise	Enterprise Arch.	M	H	M	-
Project	Project Mgr	L	M	M	-
K: strategy KPI, B: Business, S: Service, C: Component					
H: high, M: medium, L: low, N: neglectable					

Table 2. Weight granted for each role for different layer

II. Validate service components in building service architecture

The decision-making group is the same as in the previous step, consisting mainly of stakeholder, business manager, enterprise architect, project manager, business architect and other architects. Each role has its own specific weight in the various component layers, as shown in Table 2 (weight classification for role and layer).

The weight of each role is classified into the strongest R1(1.0) followed by R2(0.7), R3(0.4) and the weakest R4(0.1), while the weight of the preference rating is classified

Following the business component selection and validation, step 2 takes the validation results from the GDPM (group decision political model) and continues the validation process in the service component layer associated with the business component.

Once a business component is selected, the associated service components can be derived. The weight classification of the role and preference is the same as above but the weight granted to each role is different. This is because when the focus is shifted from K-B to B-S, the more strongly-weighted role will be shifted from stakeholder to architect. Before input is gathered from each individual, the architect team (mainly the enterprise and business architects) will describe each service component to the group so that each member has a knowledge base from which to express and to determine the significance of a service component from an enterprise point of view.

The semantic translation from business to service and from service to technical needs a deconstruction process to work on those selected business components so that each business component can be deconstructed into business use cases for further breakdown. Service components, derived from the business use case, are the mediation between business and technical elements; they integrate all business semantics, interfaces and technical configurations into a logical component.

Each service component is initially at an atomic level, meaning that it consists merely of nouns and verb from a single subject, verb and object in business terms. In other words, service components guide technical implementation for the interface and provide metadata for physical deployment. The service layer has its importance in mediating the upper layer of business metadata and the lower layer of information metadata to construct the coherence of enterprise integration. From an architectural perspective, service can exist in user access, application, information and infrastructure while from a run-time architectural perspective, the user interface service at the upper layer needs a semantic web to make information accessible to human and software agents on a semantic basis [6] [7].

III. Validate physical components of functional and non-functional elements

The derivation process shown in Table 3 explains how each use case is derived for service components in atomic form with linkage to component elements. Beneath this metadata mapping there is other information to specify the details of each use case, service component and component element with regard to their interface, semantic notation, business logic, data access and component configuration. Each layer's referential alignment is shown in Figure 2 (referential alignment between each layer).

1-4244-1029-0/07/$25.00 ©2007 IEEE.

Conclusion

In the service component-based architecture, interoperability within K-B-S-C is the key in building the skeleton of enterprise integration in a vertical direction. To cope with this objective, we apply both a political model and a rational model to validate the components in each layer. The political model applies group decision-making while the rational model applies parsing analysis which will be introduced in a separate literature; both models work on component validation with different focus of scope. Political model is the foundation of rational model as the nature of business implementation started from group decision which cannot be replaced by machine or other rules, without the human effort and validation, all the analysis and implementation will be in vain.

In role-based group decision-making, each role has a different weight in each layer within B-S-C and each group member has a different preference in their component selection. We use the political model, starting with the business components. We repeat the political model in a use case deconstruction to derive service components and physical elements. The group members validate both service components and component elements by using the same method with differently weighted roles. The evaluation between service components and physical components is based on how significant their contribution to the business goal is from a logical and physical information component perspective; a higher degree of information aggregation of C within B-S-C gives a higher preference weight in that physical component. The multiplication of B and S significant factors represents the significance of S within the K-B-S-C transformation; using the same method, the multiplication of B and C significant factors represents the significance of C within the transformation. The objective of K-B-S-C transformation and significance factor evaluation is to try to build up the vertical enterprise skeleton and extend the key services to their peripheral surroundings in performing full service virtualization.

This research work does not addresses any micro mechanism of integration interoperability as the main focus has been on macro transformation. We are currently involved in micro mechanisms for the purpose of industry solutions; these investigations will be reported separately from this research work. In our opinion, enterprise integration ontology is feasible and decision process play critical role in the initial stage.

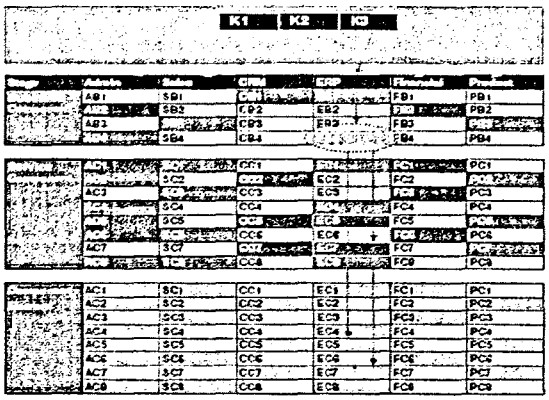


Figure 2. Referential alignment between each layer

Reference

1. Jie Lu, Guangquan Zhang & Fengjie Wu, (2005), "Web-based Multi-Criteria Group Decision Support System with Linguistic Term Processing Function", IEEE Intelligent Informatics Bulletin June 2005 Vol.5 No.1

2. Raymond Wu, (2006), "Enterprise Vertical Integration in the e-Government", Electronic Government, an International Journal (EG) ISSN (Online): 1740-7508 - ISSN (Print): 1740-7494

3. Sergey Melnik, (1999), "Generic Interoperability Framework", Distributed and Parallel Databases Journal 1999

4. DSouza,(2001)," Model-Driven Architecture and Integration Opportunities and Challenges", Kinetium

5. Guangquan Zhang and Jie Lu, (2005), "The Definition of Optimal Solution and an Extended Kuhn-Tucker Approach for Fuzzy Linear Bilevel Programming", IEEE Intelligent Informatics Bulletin November 2005 Vol.6 No.2

6. M. Staab, N. Stojanovic, R. Studer, and Y. Sure, (2001), "Semantic Portal - The SEAL approach, to appear in: Creating the Semantic Web.", MIT Press

7. O. Valkeapää and E. Hyvönen, (2006), "Semantic Annotation with Browser-based Annotation Tool", 1st Asian Semantic Web Conference

S2: Enterprise Integration Framework and Metadata

Publication: IJIIDS Journal Aug 07, Volumn 3, Number 1
International Journal of Intelligent Information and Database Systems
ISSN (Print): 1751-5858 (http://www.inderscience.com/index.php)

Publication Contact: Prof. Jason J. Jung,
Deppt. of Computer Information Engineering
Inha University, Korea
j2jung@intelligent.pe.kr



Date: Sun, 22 Apr 2007 10:47:14 +0200

From: "Ngoc-Thanh Nguyen" <Ngoc-Thanh.Nguyen@pwr.wroc.pl> Add to Addr Book Add Mobile Alert

Subject: Re: RE: Notification of IJIIDS

To: "Raymond Wu" <rwu_academic@yahoo.com>

Dear Raymond,

Your paper will be published in Vol. 1, issue 3, 2007, roughly in aug. 2007

Best regrads,
Ngoc Thanh Nguyen

From: "Jason J. Jung" <j2jung@intelligent.pe.kr> Add to Address Book Add Mobile Alert
Yahoo! DomainKeys has confirmed that this message was sent by gmail.com Learn more

To: rwu_academic@yahoo.com

CC: j2jung@intelligent.pe.kr

Subject: Notification of IJIIDS

Date: Wed, 3 Jan 2007 16:14:01 +0900

Dear Author,

We have received comments from the reviewers and decided that the paper ca be accepted for publication with minor revision.
The reviews are also enclosed for your revision.
Please improve your paper and send the revised paper until January 31th.
The revised versions should be re-reviewed for final acceptation.
I am very kindly asking you to confirm receipt of this notification.

Sincerely,

Jason J. Jung, Krzysztof Juszczyszyn, and Ngoc Thanh Nguyen
Guest Editors of IJIIDS

Jason J. Jung, Ph.D.
Department of Computer and Information Engineering
Inha University, Korea
Tel: +82 32 875 5863
http://www.intelligent.pe.kr
j2jung@intelligent.pe.kr; j2jung@gmail.com

Int. Journal of Intelligent Information and Database Systems Vol.1, No. 3

Enterprise Integration Framework and Semantic Metadata in Financial Industry

Raymond Wu

IBM Software Group, Germany
Email: raymond@de.ibm.com
Phone: 49-7031161755

Abstract: A common weakness in the interoperability between business and IT leaves IT far behind the direction business is taking. The main purpose of this paper is to conduct a research in the enterprise architecture issue as a means of addressing the vertical gap with new knowledge contributions in the vertical integration of the interconnectivity of cross-layers. To cope with this common issue, we launched enterprise integration framework in addressing the integration strategy of metadata integration between business, service and information layers, this create visibility of vertical alignment within enterprise architecture and use metadata configuration to construct the mapping between each layer. This research paper review our recent work in component based implementation and use example in financial industry to review the design process from micro component design to macro component based service virtualization. Financial engineering rely on inter-banking third party to mediate its services virtualization, there are two key processes play crucial roles in this mediation: the micro component design and macro metadata framework, the supplementary solutions of these two processes construct the enterprise integration interoperability and service virtualization which is the priority in financial industry.

Keywords: enterprise, integration, semantic, component, metadata

Biographical notes

Raymond Wu is the IT architect currently working with IBM Software Group, his expertises include, enterprise architecture, integration, metadata, service transformation, financial, telecom and database design

1 Introduction

The IT industry became pervasive in the 21st century due to its significant growth in terms of huge volume transactions and sophisticated businesses. Enterprise architecture mainly mediates between business and IT to minimize the gap by improving governance, agility and business integrity.

In our recent research, componentization, metadata and business semantic are all the foundation in enterprise integration, we proposed component based validation, analysis, design and provisioning as our approach and metadata interoperability between business, service and information as our strategy. The details of these areas and stages are shown in Figure 1.

Copyright © 2007 Inderscience Enterprises Ltd.

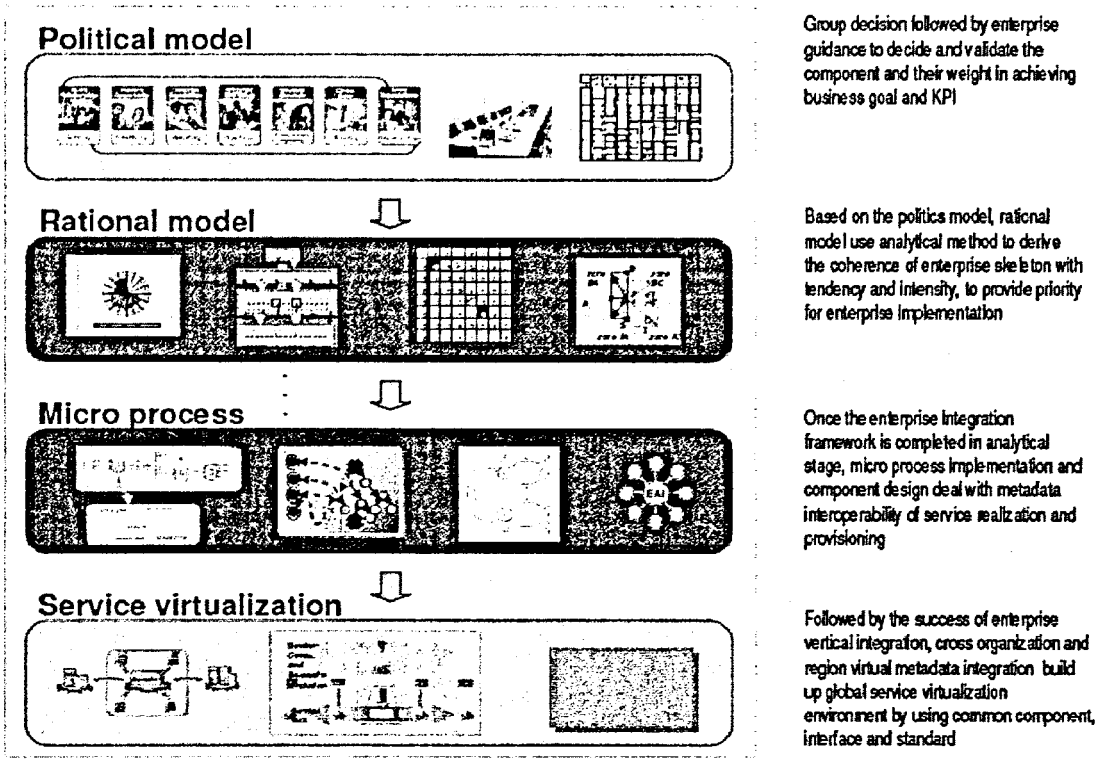


Figure 1 – Enterprise Integration Framework of Life Cycle

✧ **Decision**

We start from the decision-making process of business component selection to derive the service components and physical elements. The decision-making process uses group decision-making to take each individual opinion or preference and multiply it according to the weight specify by its role which we call "political model driven" decision-making.

✧ **Analysis**

Followed by the political model decision, we takes the validated components and continues the service component transformation we term "rational model driven" and is grounded in objectives, alternatives, consequences and optimality. In this stage we use a parsing technique to analyze the use case and break into service components at an atomic level. The parsing process calculates the frequency and dependency of each component to determine the candidacy for common service and service aggregation.

✧ **Design**

As the architecture design and configuration is defined in the analysis stage, our focus of design is the micro processes of component manufacturing, interconnectivity between components and micro process design of service provisioning; this is accomplished in a macro service environment which covers service aggregation and dimensional provisioning for a business goal, the skeleton of enterprise is formed across layers in micro processes through seamless integration.

✧ **Virtualization**

Once the vertical skeleton and service extension are built, the interoperability reflected from the business layer has the capability of integrating cross-organization and cross-geographical constraints, as fully transparent services become possible if each system or organization follows an industry standard by using metadata strategy and an industry framework.

When we apply this framework to the financial industry, first we like to investigate what is a financial project usualy involved. A typical financial project usually consists of domain implementation. architecture and shared service. The domain implementation group is divided into several teams to handle various LOB (line of business) as each team will have different areas of business domain knowledge; each team may in fact handle several business lines (i.e., use cases). It's very often the case that the individual LOB team doesn't have adequate concepts and communication to determine the common components, which include common services, common interface and standard data access and data models, we would like to review and list these common issues in enterprise integration with our proposals of resolution. this is shown in Figure 2.

i) Too many integration points cause implementation silos
ii) Data fragmentation with duplicate process without aggregation strategy (Curtis, K., Foster, P. W. and Stentiford, F., 1999)
iii) Common area is invisible and unable to reuse in a shard model
iv) Lack of priority and focus of strategy slow down business responsiveness
v) Weak enterprise interoperability cause high risk in industry transformation

In the recent requirements. the conventional credit card service doesn't maintain a robust "Fraud detection policy" at account level, so many transactions become invalid, the agility and responsiveness of making a proper change is still weak and many consumers suffer the serious issues that their transaction need be rolled back. For this reason, we launched add-on service component of Fraud Policy service to replace the conventional practice and dynamically retrieve card holder profile and the applied fraud policy to determine the authorization approval through inter-banking, so each account holder can have their security profile and preference for the card issuer or dealers to determine the individual fraud detection based on the transaction behavior and customer profile.

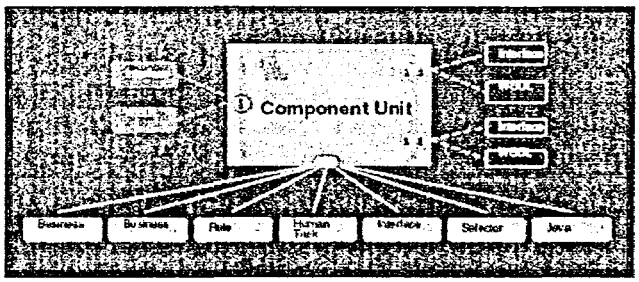


Figure 2. Service component architecture

2. Literature review

Financial industry is one of the high complexity industry among various sectors, and the severity is among those high risk areas. To eliminate the risk, our assumption is; the higher occupancy of common services within enterprise, the less suffering can be caused in each IT and business transformation, as technology and business innovation and transformation create opportunity while it may impact enterprise seriously if the transformation foundation is not solid, when they are enforced to transform their business or IT to be survival in next stage, weakness in enterprise skeleton and interoperability supported by common component may cause high risk in each transformation. According to L. Yu et al., An obstacle to software reuse is the large number of major modifications that frequently have to be made as a consequence of dependencies within the reused software components. Component base implementation is the foundation in coping with this bottleneck (DSouza., 2001) (L. Yu, 2005).

To cope with these issues, we followed the enterprise integration strategy in our recent research which includes, validation of business and service component. service analysis and transformation, service design, information aggregation and service provisioning. Within the enterprise integration life cycle,

component validation and analysis, which we introduced in earlier research papers is the foundation in component design and industry service virtualization.

3. Enterprise integration framework

Among these four stages, two key processes were used to achieve this integration objective. The first process relates to how the enterprise goal may be achieved by impacting business process and its service components level, which we call "business semantic." This process extends the concept of "separation of concerns" of service-oriented enterprises to impact all of the above elements, it decompose the business block into a business component; move forward to a service component through the business modeling process; the reason componentization needed to be set first because visibility and traceability (M. Lytras, 2006). The second process follows one level below the first one and is called "information metadata." It constitutes the repository of metadata storing information and its process of moving information between service and data; it thus supports the upper layers by repository synchronization.

4. Key strategic framework

As mentioned above, the approaches being used in this paper are:

✧ Four stages framework in the enterprise integration life cycle

Political validation, rational validation, micro process design and service provisioning are all the key stages in enterprise integration life cycle, as enterprise interoperability is built on the foundation of a validation process of decision, design and provisioning stages. A full virtualization environment, which is our objective, needs to be built on top of the foundation of component design and service transformation process in order to apply the vertical enterprise integration approach to this final stage which is the goal of our research work. The standardization of metadata technology and common services and components are all requirements within this framework (Tannenbaum. , 2001) (Dublin Core. .1997) (Maedche, S. Staab, N. Stojanovic, R. Studer, and Y. Sure., 2001) .

✧ Two process in building vertical integration direction

"Business semantic injection" which was working on the gap between strategic direction, business, process and service; and "information metadata" which is based on the application of existing metadata theory and repository solution (Lassila. , 1998). Between these two processes, the SOA which has been mature in the industry, plays cohesive mediation between semantic and metadata was service repository. The integrated solution of coherence of approaches which form the basis of this paper plays skeleton in vertical integration addresses the enterprise gap and is expected to provide further insights into micro process integration across enterprise layers (M.Gruninger, 2005).

5. Interoperability model in this framework

By using a mathematical model, we take the output from each layer denoted by a capitalized alphabetical letter (D: direction, B: business, P: process, S: service, I: information)

$\mathbf{D} = \sum dn,\ \mathbf{B} = \sum bn,\ \mathbf{P} = \sum pn,\ \mathbf{S} = \sum sn,\ \mathbf{I} = \sum in$

The conventional vertical integration can be expressed by the following union -

$\tilde{O}output = \mathbf{D} \cup \sum (\mathbf{B} \cup \mathbf{P} \cup \mathbf{I}) \cup \mathbf{S}$ in any phase

This means integration between business, process and information is a "scrambled" type without micro visibility; the vertical integration has no chance to occur at the semantic level and there is neither micro integration extendable to direction layer nor linkage to the service layer for non-service-oriented development. Moreover, there is no driving discipline for its output, as any small change in strategic direction can cause a macro change without micro insight.

The ontology we are seeking can be expressed by the following Cartesian product –

$\tilde{O}output = \sum \tilde{O}n = \sum(Bn \times Sn \times In)$ in decision and analysis stages

$\tilde{O}output = \sum \tilde{O}n = \sum(\sum(bn \times sn \times in))$ in enterprise micro design stage

$\tilde{O}output = \sum \tilde{O}^{i}n = \sum B^{i}n \times \sum S^{i}n \times \sum I^{i}n = \sum(\sum b^{i}n \times \sum s^{i}n \times \sum i^{i}n)$ in enterprise external integration

$\tilde{O}output = \sum \tilde{O}^{i}n = \sum(\sum(b^{i}n \times s^{i}n \times i^{i}n)) = \sum(B^{i}n \times S^{i}n \times I^{i}n)$ in industry virtualization stage

This means in decision and analysis stage, the alignment interoperability B × S × I is formed with micro referential integrity occurs in each layer with a loosely-coupled alignment between b-s-i, where D and P are no more exist as they became part of B and S individually, the reasons are . The ontology allows architects to apply d-b as a business-driven top-down approach or p-s as a service-driven bottom-up approach and i is simply independent; however, metadata is needed to coordinate the service layer by repository. The concatenation between B-S-I in micro process forms a reuse foundation within enterprise and industry, the standard metadata framework within industry share the repository for those common components (business, service and information) for enterprise to implement and integrate within the industry, so the service virtualization become possible. Micro process interoperability is the key in component based integration, and this is shown in Figure 3.

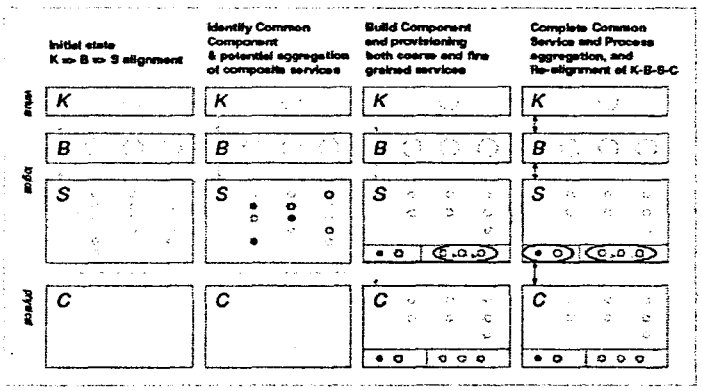


Figure 3 – Enterprise Integration Framework of K-B-S-C layers

The micro process integration was initiated by the domain context analysis process. H. Lee & J. Lee in their domain analysis - "Analyzing Business Domain: methodology and repository system" pointed out, once sets of use case scenarios are formulated, the next stage will be the extraction of verbs and nouns for identifying domain objects. This semantic analysis concept was applied in supporting the theory of vertical integration, so the analysis of the scenario and the extraction of noun and verb form the common components in process, object and interface as domain asset (H. Lee and J. Lee, 2006). The semantic analysis extract noun and verb from atomic services and form lists like this, in the semantic repository, noun list store customer name, id, phone, account number, status and verb list store, open account (type), check non-exist (db), create account (type), credit check (code). Once the semantic analysis work is done and both verb and noun list are stored in semantic repository, an asset validation of common objects need be done before further development in vertical (downward) direction, asset validation is a process for duplicate check, merge and final decision before all of these common objects become valid to stay in semantic repository for reuse. The three different types of repositories (metadata, services and semantics) collaborate each other; the strategy of "enterprise vertical integration" was based on the processes which build up these repositories in early stage, and the methods in achieving inter-operability between repositories in later stage, the fact that all of the organizations share the same service means they are all managed under the same rules which were developed using the top-down approach. As all of the

organizations share the same service means they are all managed under the same rules which were developed using the top-down approach. As all of the organizations share a common protocol to communicate their semantic rules, the integration effort can be eliminated and interoperability can be enhanced through this topology once the semantic communication has been fully developed (C. Wagner, K. Cheung, R. Ip and S. Bottcher, 2006) (G. Moynihan and R. Batson, 2006).

Further development of service component plays a key role in linkage between business and IT based on what was built in each stages. Each service component was a unique common component which performs a well-defined, specific task inside a particular business area: the critical process of "semantic injection" and "metadata aggregation" enabled the service component's capability in mediating the inter-operability of each vertical element in coping with each business change or adds-on in a high speed development environment. The governance of operability between business and IT need to configure existing services within a newly configured active map in each

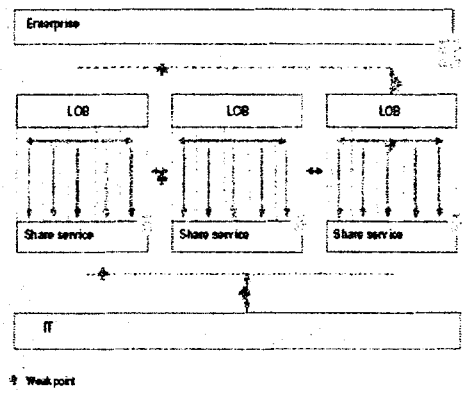


Figure4. Project architecture common issue

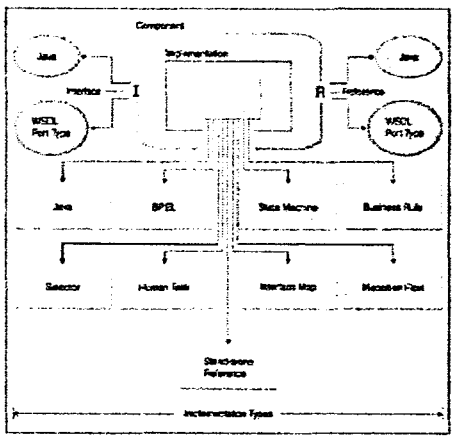


Figure 5. Service component design

change, this need to be done by using graphic mode, however in case that existing service component cannot provision full function of a new business requirement and need to develop new service component, this need be done in a design mode with linkage to business process and metadata repository (Rust, G. , 1998).

6. Component validation and analysis

Financial industry distinguished itself from other industry as time to market is critical, higher security requirements is mandatory, and inter-banking service need agility in mediating different parties into a common service infrastructure, for this reason, we apply SCA (service component architecture) shown in Figure 4, to deal with component design and interconnectivity in micro mechanism. Common service is a top-down approach follows the nature of business implementation direction.

Component validation and analysis consists of steps in validation of business components, service components, and physical components, common component parsing analysis, service aggregation parsing analysis, and implementation priority and skeleton. It will be followed by component design shown in Figure 5.

From the technical perspective the idea was to eliminate the integration effort by using the "virtual organization" concept, from the organization perspective, this enhances the flexibility of service provisioning and the agility between organization and service; thus, the interoperability across organizations is increased.

The realization of this top-down vertical integration approach requires robust mediation to support the inter-communication between component-component and component- repository, this is shown in Figure 6. In other words, the concept needs a macro framework and physical support in micro integration service bus such as the middleware in realization of run-time services.

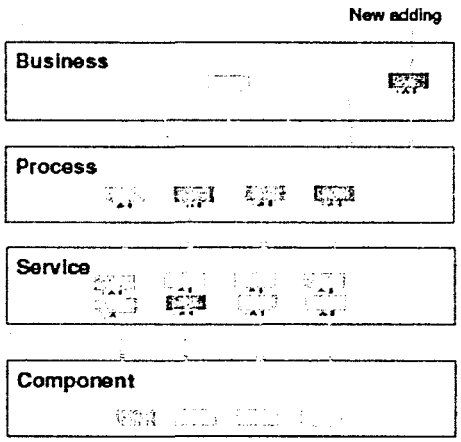


Figure 6. Top down component implementation

7. Component design in financial example

The micro process in component design follow the top-down direction and take the decomposition output to derive the service component. We use the inter-banking example of credit card industry to demonstrate the transaction between card holder (payer), merchant (payee), bank (transaction handler) and credit card authority (inter-banking). We decompose the business component of credit card transaction into several use cases: 1) Consumer contact merchant to encrypt and decrypt security keys, and request for certificates. 2) Payment is transferred from merchant to acquirer (the merchant's bank), and then to issuer through inter-bank network to authenticate the payment information. 3) Acquirer notifies merchant about the status for merchant to fill the order. 4) Merchant request acquiring bank regarding settlement. 5) Issuer charges the consumer's credit card and notifies card holder. 6) Consumer pay fee, acquirer then credits merchant for the exact fee.

The conventional design doesn't have the dynamic control to have customer insight while the issues were resolved in component based design. This is the service transformation and provisioning example which mediate the enterprise components in forming the skeleton. Service components derived from business use case are the mediation between business and technical element which integrate all business semantic, interfaces and technical configuration into a logical component, initially all the service components are in atomic level (M. Staab, N. Stojanovic, R. Studer, and Y. Sure, 2001) (O. Valkeapää and E. Hyvönen, 2006). In the component validation and analysis research, we applied group decision model in validation process, so business component with its associated service components are selected and validated through decision. We further detailed the rational model in analyzing common component and aggregated services by using use case parser to determine enterprise component in forming the skeleton of integration. Aggregation is a kind of service normalization in grouping all atomic service element from fine-grained to coarse-grained component.

We also introduced the component design and interconnectivity for how the component communicate each other and how the micro process of business semantic be derived into each component. The component element is the physical element that mediates service and information, the design of service components follows our previous strategy, by using business semantic injection to take the context into the semantic and micro processes, so that each component has its contract level responsibility in provisioning a subset of service, and the interface helps the coordination between

components and metadata build the foundation of interoperability within the service layer and between layers.

- ✧ At the very beginning of a use case, the first event triggers service Φ_1

  $\varepsilon^1 : r_0 \times \Phi_1 \rightarrow r_1$

- ✧ When the service event Φi ($i <> 1$) is read and triggers the service component, we express the state transformation as follows:

- ✧ $\varepsilon^i : r_{i-1} \times \Phi_i \rightarrow r_i$, we further derive the transformation as follows:

  $\varepsilon^{i+1} = (r_i) \times \Phi_{i+1} \rightarrow r_{i+1} = (r_{i-1} \times \Phi_i) \times \Phi_{i+1} \rightarrow r_{i+1}$

- ✧ Which is in equilibrium to $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$, and derives $\varepsilon^{nm} : r_n \times (\Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m) \rightarrow r_{m\ (n<m)}$, we call this the "semantic event aggregator."

- ✧ As the final step of a use case, the final event trigger service Φ_n

  $\varepsilon^n : r_{n-1} \times \Phi_n \rightarrow r_n$. Φ_n is a terminator, an example of which is the settlement of a daily transaction.

- ✧ The string $\Phi(r_n , r_m) = \Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m$ is a path from r_n to r_m, we call "service route" and $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$ a "composite service" between r_{i-1} and r_{i+1}, we use symbol $\Phi_{i,\ i+1}$ to denote composite service

- ✧ When frequency $F(\Phi_{i,\ i+1}) = F(\Phi_i \times \Phi_{i+1})$ is higher than threshold σ, we aggregate the concatenated alphabet (part of the string) $\Phi_i \times \Phi_{i+1}$ into $\Phi_{i\ ,\ i+1}$, this imply the original state r_i between r_{i-1} and r_{i+1} will no more exist for this composite service.

The foundation of this skeleton is interconnectivity between service component by using common component and aggregated component. We further develop the service virtualization on top of this interoperability skeleton, which we call enterprise vertical integration, in extending the integration across industry and geographical areas, this is the virtualization service in horizontal direction, once industry of different regions build up their semantic interoperability in three metadata repositories (business, service and information), they can synchronize their business, operation and strategy, so that the limitation of business, geographical area, products, application, and data structure become transparent, this is what we call service virtualization. Once the vertical skeleton and service extension are built, the interoperability reflected from the business layer has the capability of integrating cross-organization and cross-geographical constraints, as fully transparent services become possible if each system or organization follows an industry standard by using metadata strategy and an industry framework. There are three steps to reach service virtualization; (J. Hunter, 2001)

a). To identify the unique source of truth within enterprise, and use its metadata within business, service and information as a corporative standard, so when other systems and applications consolidated into enterprise repository, there is a standard and guideline to follow.

b). Vertical enterprise integration, as when other systems consolidated into the enterprise, there will be some metadata out of scope and need further integration effort to streamline the vertical integration.

c.) Service virtualization is the objective of our enterprise integration, we need to make sure our repository holding enough and accurate information within industry standard to integrate enterprise to external.

8. Conclusion

The concept of "enterprise vertical integration" covered many aspects of technology and knowledge which we already discussed, the issue was from misalignment of business and IT, so the business context and semantics cannot effectively impact IT development and cause all development, change and maintenance very difficult.

The nature of "enterprise vertical integration" differentiate itself from traditional integration from what we saw was not "architecture layer", instead we saw how business context and semantics transform

into service component and communicate with metadata through micro componentization process, this transformation totally change our concept from previous layer-to-layer integration to components' collaboration, from the A2A (application to application), B2B (business to business), D2D (data to data) to the interoperability of semantic-service-metadata repository, the new vertical integration follow the nature of the business implementation, so each new development or change of business easily identify the impacted elements for either configuration change or producing new component.

The lack of interconnectivity between context, semantics, service and data is the tactical issue with the clear implication that enterprise architecture needs a robust micro-mechanism of semantic messaging and metadata to coordinate across layers. In this way the strategic direction will then have an opportunity to impact the business process to streamline successful services and information components.

Financial industry is facing a big challenge within enterprise integration as the service concept is moving, global policy and regulation is changing and the high demands in transforming the system silos to an industry and global standard is not an easy task.

We gave the example of a typical issue in credit card scenario for how to narrow down the root issue within a small scope as a service component while maintaining its alignment to both business and technical components, this service component architecture speed up the change and enhance dramatically the integrity and responsiveness. An example of service provisioning would be the case where several key industry vendors have built up an industry framework for service providers in that particular industry to follow, starting from business domain analysis and working through to component deployment. So each service provider benefits from the industry common services and common interface, particularly those B2B, B2C, merger and acquire cases. Any business activity becomes easier when the various parties involved have a common understanding and consensus on business terms, service definitions, interfaces and component specification, and these become transparent and common practices (O. Zimmermann, P. Krogdahl and C. Gee, 2004).

Reference

(Curtis, K., Foster, P. W. and Stentiford, F., 1999) - "Metadata - The Key to Content Management Services", Proceedings of the Meta-Data '99, The third IEEE Meta-Data conference, April 6-7, 1999, Bethesa, Maryland, USA

(DSouza., 2001) - " Model-Driven Architecture and Integration Opportunities and Challenges", Kinetium

(L. Yu, 2005) - "Categorization of Common Coupling in Kernel-Based Software," 43rd ACM Southeast Conference, 2005

(M. Lytras, 2006) - "The Semantic Electronic Government: knowledge management for citizen relationship and new assessment scenarios", Electronic Government, an International Journal 2006 - Vol. 3, No.1 pp. 5 – 17

(Tannenbaum. , 2001) -"Using metamodels, repositories, xml, and enterprise portals to generate information on demand", Addison Wesley Professional, Boston, MA

(Dublin Core. ,1997) - "Dublin Core Metadata Element Set", Version 1.1: Reference Description, http://purl.org/dc/documents/rec-dces-19990702.htm

(Maedche, S. Staab, N. Stojanovic, R. Studer, and Y. Sure., 2001) - "Semantic Portal - The SEAL approach. to appear in: Creating the Semantic Web. D. Fensel et al.", MIT Press. MA, Cambridge

(Lassila. , 1998) - "Web metadata: A matter of semantics", IEEE Internet Computing, 30-7 http://www.uie.com/

(M.Gruninger, 2005) - "Model-theoretic approaches to semantic integration", Semantic Interoperability and Integration

(H. Lee and J. Lee, 2006) - "Analyzing Business Domain: A Methodology and Repository System", Korea Advanced Institute of Science and Technology

(C. Wagner, K. Cheung, R. Ip and S. Bottcher, 2006) - "Building Semantic Webs for e-government with Wiki technology", Electronic Government, an International Journal 2006 - Vol. 3, No.1 pp. 36 - 55

(G. Moynihan and R. Batson, 2006) - "Development of a matrix methodology for database reengineering and improvement: an e-government case study", Electronic Government, an International Journal 2006 - Vol. 3, No.2 pp. 190 - 203

(Rust, G. , 1998) - "Metadata: The Right Approach. An Integrated Model for Descriptive and Rights Metadata in E-commerce", D-Lib Magazine, July/August 1998. http://www.dlib.org/dlib/july98/rust/07rust.html

(M. Staab, N. Stojanovic, R. Studer, and Y. Sure, 2001) - "Semantic Portal - The SEAL approach. to appear in: Creating the Semantic Web.", MIT Press

(O. Valkeapää and E. Hyvönen, 2006) - "Semantic Annotation with Browser-based Annotation Tool", 1st Asian Semantic Web Conference

(J. Hunter, 2001) - "MetaNet - A Metadata Term Thesaurus to Enable Semantic Interoperability Between Metadata Domains", Journal of Digital Information, Volume 1 Issue 8

(O. Zimmermann, P. Krogdahl and C. Gee, 2004) - "Elements of Service-Oriented Analysis and Design", IBM Academy

S3: Enterprise Integration in Metadata Environment

Publication: JOT Journal ISSN 1660-1769

JOURNAL OF OBJECT TECHNOLOGY

The Authority on Objects and Components

http://www.jot.fm/issues/issue_2007_11/

NOVEMBER - DECEMBER 2007 VOL. 6. No. 10

PREVIOUS ISSUE ◇ NEXT ISSUE

Enterprise Integration in Metadata Environment

By Raymond Wu

Enterprise integration minimize the gap between business and IT to improve governance, agility and integrity. Two key processes were used to achieve this objective, the first process relates to how the enterprise goal can be achieved by impacting business process and its service components level, which we call "business semantic" or macro architecture. The second process follows one level below the first one called metadata mediation which called micro process, it constitutes the repository of metadata storing aggregated information and symbolic service information

Enterprise Integration in Metadata Environment

Raymond Wu
IBM Software Group
raymond@de.ibm.com

Abstract

Metadata play crucial role in enterprise interoperability between business, service and information layers. Metadata in different form, such as the abstract of system structure, aggregation of information, repository and semantic mediation play different role to achieve the integration interoperability, so each type of metadata contribute their value to enterprise integration. As misalignment within enterprise architecture has been ranked one of the top issues in recent 10 years, to cope with this common issue, we launched metadata based integration framework to enhance the visibility of enterprise alignment and use metadata configuration to construct the mapping between layer.

Keywords: enterprise integration, metadata interoperability, framework, alignment

Introduction

Enterprise integration minimize the gap between business and IT to improve governance, agility and integrity. Two key processes were used to achieve this objective, the first process relates to how the enterprise goal can be achieved by impacting business process and its service components level, which we call "business semantic" or macro architecture. The second process follows one level below the first one called metadata mediation which called micro process, it constitutes the repository of metadata storing aggregated information and symbolic service information. Although there is dependency of micro process on macro process, the micro process integration should be initiated from the domain context analysis which proposed by H. Lee & J. Lee, so the three different forms of repositories (metadata, services and semantics) collaborate each other in creating the enterprise alignment between each layer in supporting macro architecture. Enterprise interoperability using common services is our focus, and metadata strategy is mandatory for common service, the fact that all of the organizations share the same service called common service which means they are all managed under the same rules which were developed using the top-down approach [1], [2].

In our recent research, componentization, metadata and business semantic are all the foundation in enterprise integration, we proposed component based validation, analysis, design and provisioning as our approach and metadata interoperability between business, service and information as our strategy.

In our enterprise integration roadmap, we start from the decision-making process of business component selection to derive the service components and physical elements. In decision stage, group decision take each individual opinion and their role's weight under "political model driven" to validate either enterprise component or component required by a business scenario. Followed by the political model decision, we takes the validated components and continues the service component transformation, we term "rational model driven" and is grounded in objectives, alternatives, consequences and optimality. In this

stage we use a parsing technique to analyze the use case and break into service components at an atomic level. We therefore, work on the micro process of component level interoperability and their aggregation. So once the vertical skeleton and service extension are built, the interoperability reflected from the business layer has the capability of integrating cross-organization and cross-geographical constraints, as fully transparent services become possible if each system or organization follows an industry standard by using metadata strategy and an industry framework.

Literature review

C. Foste et al. indicated metadata has a key role to play in allowing systems to be built and automated. A management system composed of multiple distributed components implements a service such as content validation, transcoding, search or generation; and relies on metadata information to achieve success [3]. In our metadata strategy, service components derived from business use case are the mediation between business and technical element which integrate all business semantic, interfaces and technical configuration into a logical component, initially all the service components are in atomic level [4] [5] and these are achievable by metadata. This explain why we build the enterprise integration on top of metadata foundation. S. Melnik (1999) proposed a universal interface which avoids silos in models and languages, the generic interoperability framework developed to facilitate integration of heterogeneous information systems [6] and this is the foundation of our common services approach. Based on this common framework, we can build the component of both functional and non-functional by using industry standard.

DSouza specified the standard and common component as followed - to manage models of large scale, a coherent modeling architecture is required, with standard ways to describe shared concepts, rules, patterns, frameworks, mappings, and generators. [7]. M. Schmalz argued that database schema and application code can be efficiently derived from various types of schema representations, particularly the relational model, and supports comparison of a wide variety of schema and code constructs [8]. According to L. Yu et al., an obstacle to software reuse is the large number of major modifications that frequently have to be made as a consequence of dependencies within the reused software components. Component base implementation is the foundation in coping with this bottleneck [9].

Thus, the determination of Common Services and Composite Services is a use cases driven parsing process, the purpose of the parsing iteration process is trying to find out the common areas and dependency between services. The determination of common component and service aggregation, as mentioned, is through the parsing process using use case driven model.

Integration Framework

To cope with the enterprise integration issue, our priority is a parallel process between architecture strategy and implementation roadmap, we proposed four steps META (Method-Evaluation-Technique-Application) approach, the approach applys the strategy (Method) of vertical integration to business decomposition (Evaluation), so once the component is evaluated and validated, we can apply parsing and aggregation (Techniques) in component based implementation and to achieve enterprise and industry virtualization (Application). Among these four stages, two key processes were used to achieve this integration objective. The first process relates to how the enterprise goal can be achieved by impacting business process and its service components, which we call "business semantic."

This process comply with the concept of "separation of concerns" of service-oriented enterprises to impact all of the above elements, it decompose the business block into business component; move forward to a service component through the business modeling process; the reason componentization needed to be the first priority is because visibility and traceability requrements [10]. The second process follows the first one is called "metadata collaboration". It constitutes the repository of metadata storing information and its process in coordinating message between service and data; it thus supports the upper layers, which is the first process, by repository synchronization. The details of these areas and stages are shown in Figure 1.

As mentioned above, the approaches being used in this paper include: i) four stages framework in the enterprise integration life cycle, strategic framework, political validation, rational validation and micro process design, and service provisioning. These are all the key stages in enterprise integration life cycle, as enterprise interoperability is built on the foundation of validation processes of decision, design and provisioning stages. A full virtualization environment, which is our objective, needs to be built on top of the foundation of component design and service transformation process in order to apply the vertical enterprise integration approach to this final stage which is the goal of our research work. The standardization of metadata technology and common services and components are all requirements within this framework [11] [12] [13]. ii) two process in building vertical interoperability which includes: "business semantic injection" which was working on the gap between strategic direction, business, process and service; and "information metadata" which is based on the application of existing metadata theory and repository solution [14]. The integrated solution of coherence of approaches plays skeleton in vertical integration addresses the enterprise gap and is expected to provide further insights into micro process integration across enterprise layers [15].

Interoperability model framework

To extend the concept of metadata interoperabilit by using a mathematical model, we take the output from each layer denoted by a capitalized alphabetical letter (D: direction, B: business, P: process, S: service, I: information). In decision and analysis stage, the alignment interoperability between B, S and I is formed with micro referential integrity occurs in each layer with a loosely-coupled alignment between b-s-i, where D and P are no more exist as they became part of B and S individually, the reasons are: 1) the ontology allows architects to apply d-b as a business-driven top-down approach or p-s as a service-driven bottom-up approach, however metadata is needed to coordinate the service layer by metadata repository. 2) the concatenation between B-S-I in micro process forms a reuse foundation within enterprise and industry, the standard metadata framework within industry share the repository for those common components (business, service and information) for enterprise to implement and integrate within the industry, so the service virtualization become possible. Micro process interoperability is the key in component based integration, and this concept is shown in Figure 2.

The micro process integration was initiated by the domain context analysis process, the reason is each domain analysis produces one or many service dimention which requires support from metadata and comonent coherence, this differentiate our approach from conventional approach as usually in convntional design, either the micro process is mssing or they are not appear in the early stage.

The semantic analysis concept was applied in supporting the theory of vertical integration, so the analysis of the scenario and the semantic extraction of noun and verb form the common components in process, object and interface as domain asset [16]. The semantic analysis extract noun and verb from atomic services and form lists like this, in the semantic repository, noun list store customer name, id, phone, account number, status and verb list store, open account (type), check non-exist (db), create account (type), credit check (code). Once the semantic analysis work is done and both verb and noun list are stored in semantic repository, an asset validation of common objects need be done before further development in vertical (downward) direction, asset validation is a process for duplicate check, merge and final decision before all of these common objects become valid to stay in semantic repository for reuse. The three different types of repositories (metadata, services and semantics) collaborate each other; the strategy of "enterprise vertical integration" was based on the processes which build up these repositories in early stage, and the methods in achieving inter-operability between repositories in later stage, the fact that all of the organizations share the same service means they are all managed under the same rules which were developed using the top-down approach. As all of the organizations share a common protocol to communicate their semantic rules, the integration effort can be eliminated and interoperability can be enhanced through this topology once the semantic communication has been fully developed [17], [18]. Each service component was a unique common component which performs a well-defined, specific task inside a particular business area; the critical process of "semantic injection" and "metadata aggregation" enabled the service component's capability in mediating the inter-operability of each vertical element in coping with each business change or adds-on in a fast development environment. The governance of operability between business and IT need to configure existing services within a newly configured active map in each change which is the selected business components in achieving a particular business actitivity, this need to be done by using graphic mode, however in case that existing service component cannot provision full function of a new business requirement and need to develop new service component, this need be done in a design mode with manual linkage to business process and metadata repository.

Conclusion

The concept of "enterprise vertical integration" covered many aspects of technology and knowledge which we already discussed, the issue was from misalignment of business and IT, so the business context and semantics cannot effectively impact IT development and cause all development, change and maintenance very difficult. Metadata coordiation in component based design is the first step we like to propose to achive both of our architecture strategy and integration roadmap. The lack of interconnectivity between context, semantics, service and data is the tactical issue with the clear implication that enterprise architecture needs a robust micro-mechanism of semantic messaging and metadata to coordinate across layers. In this way the strategic direction will then have an opportunity to impact the business process to streamline successful services and information components.

The nature of "enterprise vertical integration" differentiate itself from traditional integration from what we saw was not "architecture layer", instead we saw how business context and semantics transform into service component and communicate with metadata through micro componentization process, this transformation totally change our concept from previous layer-to-layer integration to components' collaboration. The enterprise integration life cycle consists validation, analysis, design and virtualization. Enterprise interoperability is built on the foundation of metadata and component based design, we applied both macro and micro

process in constructing the foundation, we further aggregate the component into common component and aggregation services. This encompasses component-based design and interconnectivity, transformation of service aggregation through rule-based design, service provisioning in a particular dimension and service virtualization.

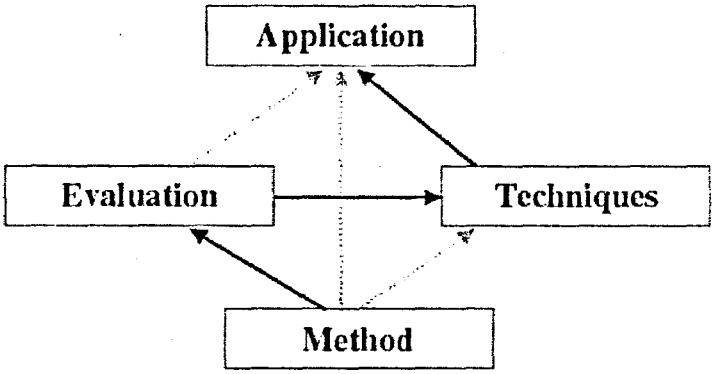


Figure 1 – META approach

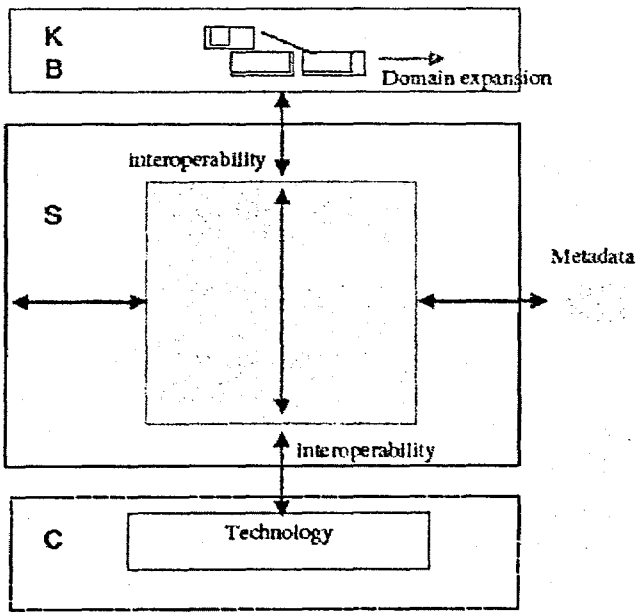


Figure 2 – Enterprise Integration Framework of K-B-S-C metadata alignment

Reference

1. Heeseok Lee and Jae Lee (2006), "Analyzing Business Domain: A Methodology and Repository System", Korea Advanced Institute of Science and Technology
2. C. Wagner, K. Cheung, R. Ip, S. Bottcher, (2006), "Building Semantic Webs for e-government with Wiki technology", Electronic Government, an International Journal 2006 - Vol. 3, No.1 pp. 36 - 55
3. C. Foster, P. Stentiford, (1999), "Metadata - The Key to Content Management Services, Proceedings of the Meta-Data", The third IEEE Meta-Data conference
4. M. Staab, N. Stojanovic, R. Studer, and Y. Sure, (2001), "Semantic Portal - The SEAL approach, to appear in: Creating the Semantic Web.", MIT Press
5. O. Valkeapää and E. Hyvönen, (2006), "Semantic Annotation with Browser-based Annotation Tool", 1st Asian Semantic Web Conference
6. Sergey Melnik, (1999), "Generic Interoperability Framework", Distributed and Parallel Databases Journal 1999
7. DSouza,(2001)," Model-Driven Architecture and Integration Opportunities and Challenges", Kinetium
8. M. Schmalz, 2003, "EITH – A Unifying Representation for Database Schema and Application Code in Enterprise Knowledge Extraction" , 22nd International Conference on Conceptual Modeling
9. L. Yu, (2005), "Categorization of Common Coupling in Kernel-Based Software," 43rd ACM Southeast Conference, 2005
10. M. Lytras, (2006), "The Semantic Electronic Government: knowledge management for citizen relationship and new assessment scenarios", Electronic Government, an International Journal 2006 - Vol. 3, No.1 pp. 5 – 17
11. Tannenbaum. Metadata Solutions: (2001), "Using metamodels, repositories, xml, and enterprise portals to generate information on demand", Addison Wesley Professional, Boston, MA
12. Dublin Core. (1997), "Dublin Core Metadata Element Set", Version 1.1: Reference Description, http://purl.org/dc/documents/rec-dces-19990702.htm
13. Maedche, S. Staab, N. Stojanovic, R. Studer, and Y. Sure. (2001), "Semantic Portal - The SEAL approach. to appear in: Creating the Semantic Web. D. Fensel et al.", MIT Press, MA, Cambridge
14. Lassila. (1998), "Web metadata: A matter of semantics", IEEE Internet Computing, 30-7 http://www.uie.com/
15. Michael Gruninger (2005), "Model-theoretic approaches to semantic integration", Semantic Interoperability and Integration
16. Heeseok Lee and Jae Lee (2006), "Analyzing Business Domain: A Methodology and Repository System", Korea Advanced Institute of Science and Technology
17. C. Wagner, K. Cheung, R. Ip, S. Bottcher, (2006), "Building Semantic Webs for e-government with Wiki technology", Electronic Government, an International Journal 2006 - Vol. 3, No.1 pp. 36 - 55
18. G. Moynihan, R. Batson, (2006), "Development of a matrix methodology for database reengineering and improvement: an e-government case study", Electronic Government, an International Journal 2006 - Vol. 3, No.2 pp. 190 - 203

S4

S4: Case study of micro component design and industry integration

Publication: (IJIIM) ISSN (Online): 1750-0036 - ISSN (Print): 1750-0028
International Journal of Intercultural Information Management
http://www.inderscience.com/browse/index.php?journalCODE=ijiim

Publication Contact: Prof. Chang T. Hsieh
McCarty Distinguished Professor in MIS
Editor-in-Chief, International Journal of Intercultural Information management, www.inderscience.com/ijiim
University of Southern Mississippi
Hattiesburg, MS 39406
601-266-4641, Chang-tseh.Hsieh@usm.edu



From: "Chang-tseh Hsieh" <Chang-tseh.hsieh@usm.edu> Add to Address Bool

To: "'Raymond Wu'" <rwu_academic@yahoo.com>

Subject: RE: IJIIM6053

Date: Sun, 17 Jun 2007 17:31:57 -0500

Dear [illegible] Wu,

After a thorough double-blind review, I am very pleased to inform you that your revised manuscript entitled " Case study of micro component desig and industry integration," (IJIIM6053) is scheduled for publication in a forthcoming issue of the International Journal of Intercultural Information Management), if you fulfill the following six required actions.

Chang T. Hsieh, Ph.D.
McCarty Distinguished Professor in MIS
Editor, Journal of International Technology and Information Management
<http://www.iima.org/New%20Pages/JITIM.html>
Editor-in-Chief, International Journal of Intercultural Information management, www.inderscience.com/ijiim
University of Southern Mississippi
118 College Drive, #5178
Hattiesburg, MS 39406

International Journal of Intercultural Information Management

REVIEWER COMMENTS

IJIIM Paper Code: 6053

The paper is rated using the following dimensions. Comments are also provided to help the author(s).

	Not at all		Somewhat			Completely	
1. This paper makes a new or substantial contribution to the literature in its own area?	1	2	3	4	5	6	7 X
Comments:							
2. The author(s) accomplished the objectives stated in their research.	1	2	3	4	5	6	7 X
Comments:							
3. New insights (learn new things) were gained from reading the paper.	1	2	3	4	5	6	7 X
Comments **Important: new point of view of the micro component-based service virtualization.**							
4. The methodology used is appropriate?	1	2	3	4	5	6	7 X
Comments:							
5. Paper is stylistically appropriate for IJIIM (Clear, concise, and absent of clichés)	1	2	3	4	5	6	7 X
Comments							
6. The title and abstract are appropriate for the paper. References are sufficiently complete.	1	2	3	4	5	6	7 X
Comments:							
	Poor-Not Appropriate or not Publishable		Average-Good/ Needs modifications and work			Very Good/ Publishable with little or no work	
7. What is your overall assessment of the paper?	1	2	3	4	5	6	7 X
Comments							

Overall Recommendation

Overall recommendation for including the paper in IJIIM:

[X] Be accepted by IJIIMas it stands
[ ] Be accepted with minor modifications (be specific)
[ ] Be accepted with major modifications (please be specific)
[ ] Be rejected?
[ ] Be recommended for submission to a more appropriate journal? (please be specific)

IJIIM - International Journal of Intercultural Information Management

Case Study of Micro Component Design and Industry Integration

Raymond Wu

IBM Software Group
Email: rwu_academic@yahoo.com
Address: IBM CDL, Diamond Bldg, ZGC Software Park #19
Dongbeiwang West Road #8, Haidian District, Beijing P.R.C.100094

Abstract: *According to our three steps plan, enterprise integration requires an ontology of elements, services transformation and virtualization to reach the goal. Componentization is critical and also the first step in component-based service design; it identifies each component in business, service and physical layers to build up the alignment and inter-layer interoperability. This research paper focuses on identification and specification of components and uses examples in the industry to demonstrate how services are derived from business components, and how they are designed and produced in achieving service virtualization. This paper also reviews micro component design in order to achieve macro component-based service virtualization.*

Keywords: component design, enterprise integration, service transformation, financial industry

Biographical notes

Raymond Wu is the IT architect currently working with IBM Software Group, his expertises include, enterprise architecture, integration, metadata, service transformation, financial, telecom and database design. Previously he worked with Oracle, Citibank and Vodafone as enterprise architect.

1. Introduction

A component conforms to and provides the physical realization of a set of interfaces. First, a component is non-trivial; it is functionally and conceptually larger than a single class or a single line of code. Component validation and analysis consists of steps in validation of business components, service components, and physical components; common component parsing analysis; service aggregation parsing analysis; and implementation priority and skeleton. It will be followed by component design. Typically, a component encompasses the structure and behaviour of a collaboration of classes. Second, a component is nearly independent of other components. It rarely stands alone. A given component collaborates with other components and in so doing assumes a specific architectural context. This architectural context is driven in large part by the implementation we choose. Third, a component is a replaceable part of a system. A component is substitutable for any other component that realizes the same interfaces. This helps during development, as parts of a system can be stubbed, sketched and then replaced by mature, robust implementations. It also supports the evolution of a system even after it has been deployed, by making it possible to upgrade and evolve parts of the system independently. Fourth, a component fulfils a clear function. It is logically and physically cohesive and thus denotes a meaningful structural and/or behavioural chunk of a larger system. It is by no means an arbitrary grouping. Fifth, a component exists in the context of a well-defined architecture. From the technical perspective the idea was to eliminate the integration effort by using the "virtual organization" concept. From the organization perspective, this enhances the flexibility of service provisioning and the agility between organization and service; thus, the interoperability across organizations is increased.

Copyright © 2007 Inderscience Enterprises Ltd.

Our architecture interoperability is based on component validation, component design, micro process mediation and dimensional services. The realization of this top-down vertical integration approach requires robust mediation to support the inter-communication between component-component and component-repository. In other words, the concept needs a macro framework and physical support in micro integration service bus such as the middleware in realization of run-time services.

The financial industry distinguishes itself from other industries in that time to market is critical, higher security requirements are mandatory, and inter-banking service need agility in mediating different parties into a common service infrastructure. For this reason, we apply SCA (service component architecture) to deal with component design and interconnectivity in micro mechanism. Common service is a top-down approach which follows the nature of business implementation direction.

A typical financial project consists of domain implementation, architecture and shared services. The domain implementation group is divided into several teams to handle various LOBs (lines of business) as each team will have different areas of business domain knowledge; each team may in fact handle several business lines (i.e., use cases). It's very often the case that the individual LOB team doesn't have adequate concepts and communication to determine the common components, which include common services, common interface and standard data access and data models. We would like to review and list these common issues in enterprise integration, along with our proposals for resolution.

- Too many integration points cause implementation silos
- Data fragmentation with duplicate process without aggregation strategy [1]
- Common area is invisible and unable for reuse in a shared model
- Lack of priority and focus of strategy slow down business responsiveness
- Weak enterprise interoperability causes high risk in industry transformation

This scenario is common in the financial industry. In the recent configuration, the conventional credit card service doesn't maintain a robust fraud detection policy at the account level, so many transactions become invalid, the agility and responsiveness of making a proper change is still weak, and many consumers suffer the serious issue that their transaction needs to be rolled back. For this reason, we launched the add-on service component of Fraud Policy service to replace the conventional practice and dynamically retrieve cardholder profile, and the Applied Fraud Policy to determine the authorization approval through inter-banking. In this manner it is the security profile and preferences of the individual account holder which determine the fraud detection, based on the transaction behavior and customer profile.

2. Literature review

According to L. Yu et al., an obstacle to software reuse is the large number of major modifications that frequently have to be made as a consequence of dependencies within the reused software components. Component-based implementation is the foundation in coping with this bottleneck [2].

The financial industry is one of the most highly complex among the various sectors, and along with the complexity comes high risk. To eliminate the risk, our assumption is: The higher occupancy of common services within the enterprise, the less suffering will be caused in each IT and business transformation, as technology and business innovation and transformation, while creating opportunity, may impact the enterprise seriously if the transformation foundation is not solid. Weakness in enterprise skeleton and in interoperability supported by common components, may cause high risk in each transformation necessary to survival in the next stage. To cope with these issues, we followed the enterprise integration strategy in our recent research which includes: validation of business and service components, service analysis and transformation, service design, information aggregation and service provisioning. Our research in enterprise integration in addressing these issues has been divided into four areas: validation, analysis, design and virtualization [3]. According to P. Kruchten, a component is a non-trivial, nearly independent and replaceable part of a system that fulfils a clear function in the context of a well-defined architecture. Desmond DSouza defines vertical integration in his "Model-Driven Architecture and Integration Opportunities and Challenges," stating that vertical integration comes from different levels of abstraction

Copyright © 2007 Inderscience Enterprises Ltd.

of the same system, from physical data models, logical data models, networks, applications specifications, component assemblies, business process models and business goals and strategies [4].

Common interface is also a means to achieve common components. S. Melnik proposes a universal interface which avoids common models and languages; this is the Generic Interoperability Framework which was developed to facilitate the integration of heterogeneous information systems. Within this framework individual communication protocols, data manipulation languages and data are represented in a generic manner while preserving their ontological variety; this process indicates how a mediation infrastructure can be built using canonical wrappers, operational requirements of integration and layered architecture for component design [5]. D. Oberle, S. Staab and A. Eberhart, in their Kaon server design, show several potential sources of semantic metadata. These sources provide input for the framework regarding the semantic management of software components. This includes deployment descriptors, Web and application server configuration files, source code information or annotations and database metadata. The component metadata collector parses this information and automatically integrates it into the ontology as semantic metadata (instances). The inference engine is embedded in the application server and applies the ontology and semantic metadata for reasoning and querying. The application server's core functionality and the administration console both use the reasoning capability. The latter allows the developer to browse and query the ontology at runtime so that configurations may be queried or checked to avoid inconsistent system configurations during development and at runtime [6].

3. Inter-banking in credit card transaction

We break down the business component into the following use-case scenarios for further development. The business scenario is related to credit card transaction between cardholder, merchant and inter-banking. We deconstruct the business components into the following service use cases in sequential order, and use Figure 1 to illustrate the business scenario:

1: Buyer sends payment information to the merchant server. Merchant software must send and receive messages, encrypt and decrypt, store public and private keys, and request and receive certificates.

2: Merchant software takes payment information from the cardholder and sends it to the acquirer (the merchant's bank). The acquiring institution must receive and authenticate the payment information the merchant received from the cardholder.

3: Acquirer sends an authorization request to the issuer over the inter-bank network. Issuer sends the authorization response to acquirer.

4: Acquirer notifies merchant about the status of authorization; if the response is positive, the merchant fills the order.

5: Merchant presents the charge to the acquiring bank.

6: Acquirer sends a settlement request to the issuer.

7: Issuer charges the buyer's credit card account and at regular intervals notifies the buyer of transactions and accumulated charges.

8: Buyer pays the charges to the bank. Acquirer credits the merchant's account.

In our current research, we are focusing on the interoperability of service provisioning and virtualization. In the conventional credit card service, the inter-banking doesn't maintain a "fraud detection policy" service at the account level. Mostly this shortage is the result of conventional silo implementation and limitation of multi-site interconnectivity such that a daily program is used to update the hard-coded matrix within the whole authorization process.

The issue is with a non-component-based environment in which implementation can cause high impact on the enterprise, and may even be unachievable between different organizations having a common metadata to handle. These are shown in Figure 2, which illustrates the add-on component provisioning a new service in fraud detection as a dynamic option based on customer profile. Our newly launched add-on services include a component-based Fraud Policy service which, as explained in the Introduction, will replace the conventional practice and dynamically retrieve cardholder profile and the applied fraud policy to determine the authorization approval through inter-banking, so each account holder's security profile

Copyright © 2007 Inderscience Enterprises Ltd.

and preference for the card issuer or dealers will determine the individual fraud detection based on the transaction behaviour and customer profile. This is the service transformation and provisioning example which mediates the enterprise components in forming the skeleton. This skeleton requires aggregation and composite services, and semantic interoperability as well.

The semantic interpretation from business to service, and from service to technical, needs a deconstruction process to work on those selected business components, so that each business component can be deconstructed into business use cases for further breakdown. Service components derived from the business use case are the mediation between business and technical elements which integrate all business semantics, interfaces and technical configurations into a logical component. Initially all the service components are at an atomic level [7] [8].

Service normalization means grouping all atomic service elements from fine-grained to coarse-grained. The conventional credit card service doesn't maintain a fraud detection policy service at the account level; it is maintained as a generic rule applied to all cardholders within the entire authorization process.

Prior to the component design, the business component selection, deconstruction for service component validation and analysis are required in component-based design for enterprise integration objective. We will not detail the process as these areas were the focus of our previous research. We introduced a group decision model in the validation process, so business components with their associated service components are selected and validated through the decision process. Table 1 shows the alignment of the validated business component, the deconstructed use case, the service component and component elements. We further detailed the rational model in analyzing common components and aggregated services by using a use case parser to determine enterprise components in forming the skeleton of integration. Aggregation is a kind of service normalization in grouping all atomic service elements from fine-grained to coarse-grained components. We also introduced the component design and interconnectivity for how the components communicate with each other and how the micro process of the business semantic is derived in each component.

4. Component identification and design

The design of service components follows our previous strategy, by using business semantic injection to take the context into the semantic and micro processes, so that each component has its contract-level responsibility in provisioning a subset of services, and the interface helps the coordination between components and metadata build the foundation of interoperability within the service layer and between layers. In the following paragraph, we take the results from component design into the service level, and meanwhile we apply automata theory to build up the rule-based service aggregation at the enterprise level. As we are still at a pioneer stage in applying automata theory in service provisioning and aggregation, we understand this cannot be accomplished within a short period, and further research efforts will be undertaken following the current work [9].

The string $\Phi(r_n, r_m) = \Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m$ is a path from r_n to r_m, we call "service route" and $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$ a "composite service" between r_{i-1} and r_{i+1}, we use symbol $\Phi_{i, i+1}$ to denote composite service. When frequency $F(\Phi_{i, i+1}) = F(\Phi_i \times \Phi_{i+1})$ is higher than threshold σ, we aggregate the concatenated alphabet (part of the string) $\Phi_i \times \Phi_{i+1}$ into $\Phi_{i, i+1}$, this implies the original state r_i between r_{i-1} and r_{i+1} will no longer exist for this composite service.

From the information metadata perspective, to achieve the service provisioning in single view of the customer, we aggregate all fragmented data sources into metadata and business intelligence to build the service virtualization foundation based on the integrity between business, service and information. M. Schmalz argued that database schema and application code can be efficiently derived from various types of schema representations, particularly the relational model, and supports comparison of a wide variety of schema and code constructs [10]. Once we create insight into the customer's preference and transaction behaviour, we can effectively add high-value services on top of the information metadata, which represent the accuracy of customer, summary information and business intelligence; this is the information aggregation metadata concept. The interoperability skeleton is an example of how the micro interoperability in the service layer, aligned with the component layer in micro mechanism, provisions the macro service in business. The foundation of this skeleton is interconnectivity between service

Copyright © 2007 Inderscience Enterprises Ltd.

components by using common components and aggregated components. We further develop the service virtualization on top of this interoperability skeleton, which we call enterprise vertical integration, in extending the integration across industry and geographical areas. This is the virtualization service in a horizontal direction: once the industries of different regions build up their semantic interoperability in three metadata repositories (business, service and information), they can synchronize their business, operation and strategy, so that the limitations of business, geographical area, products, application and data structure become transparent. This is what we call service virtualization.

The transformation therefore requires an event to incur the service with the intelligence to either predict the next service or to verify that the called service is recognized. We use the state concept discussed above to mediate the business, service event and information in the "time" dimension. The state of the service layer and the business-service-information coherence are changed once we trigger an event and the service is successfully executed. The alphabet read in automata is leveraged to service component trigger and execution while the state change is leveraged to transformation within the service layer. In any component context a formal interface contract for the invocation syntax is important. The semantics issue (the meaning of parameters and so forth) has to be solved as well (domain modelling). This is crucial to any business-to-business (B2B) and dynamic invocation scenario. Such scenarios are cornerstones of the service component vision of being flexible and agile in response to the new business needs in a world of mergers and acquisitions, business transformation, globalization and other relevant factors [11].

A tool called JFLAP guides the user through several proofs of theorems that focus on transformations of languages from one form to another in which the forms are automata and grammars. JFLAP aids in automata visibility; users are given a visual representation they can interact with in addition to the formal textual representation. This alternative hands-on representation provides a means for an interactive automata theory [12]. The string $\Phi(r_n, r_m) = \Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m$ is a path from r_n to r_m, we call the "service route" and $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$ a "composite service" between r_{i-1} and r_{i+1}, we use the symbol $\Phi_{i, i+1}$ to denote a composite service.

When frequency $F(\Phi_{i, i+1}) = F(\Phi_i \times \Phi_{i+1})$ is higher than threshold σ, we aggregate the concatenated alphabet (part of the string) $\Phi_i \times \Phi_{i+1}$ into $\Phi_{i, i+1}$, this implies that the original state r_i between r_{i-1} and r_{i+1} will no longer exist for this composite service. When the business architect checks the result from semantic event aggregator - ε^m which is more an upper-layer use case scenario, service route - $\Phi_{i, i+1}$ which is in the service layer and its frequency - $F(\Phi_{i, i+1})$, if $F(\Phi_{i, i+1})$ is much higher than threshold σ, then from the business perspective the event aggregator ε^m either consists of many composite services or is business critical. This means the aggregation service is the key within business and is crucial in daily operation from service perspectives, and that fine tuning is feasible from a multi-fine-grained boundary to a coarse-grained service. We call this the "main route" of the enterprise skeleton; in other words this is the highway between A and B. Virtual coarse-grained service eliminates the boundary and leverages many fine-grained components which can be easily configured by adding or modifying the component dynamically to achieve the broader coverage of main route services. For this reason main route is one of the crucial services in enterprise integration. One example is in the telecommunications industry that incorporates an open service account, an open billing account and provisioning options for a new customer.

This constitutes a main route for "new customer service" coarse-grained service, though it is aligned with fine-grained components to support customer preference, service criteria and billing options which are branches within this main route.

The importance of determining the common component and service aggregation is in building up an enterprise skeleton of vertical interoperability between business, service and information. This task is crucial, as in the business domain analysis stage, the impact area and scope to business components are minimized in decision making. Additionally, the business component is deconstructed into service components and physical elements for top-down alignment while the common component is not simply a

Copyright © 2007 Inderscience Enterprises Ltd.

symbolic service but instead requires a bottom-up approach to work on common areas; this serves to fine tune the common components and interfaces from the perspective of enterprise integrity and performance.

Figure 3 shows how common components and composite component service aggregation form the skeleton of enterprise integration; the skeleton extends its service to the derived service and forms the interoperability with upward linkage to business and downward to physical components. The service aggregation is also important as the top-down deconstruction followed by bottom-up composition forms the skeleton of vertical interoperability and potential component-based implementation of the service process.

5. Component design and integration in the financial industry

The micro process in component design follows the top-down direction and takes the deconstruction output to derive the service component. We use the inter-banking example of the credit card industry to demonstrate the transaction between cardholder (payer), merchant (payee), bank (transaction handler) and credit card authority (inter-banking). We deconstruct the business component of credit card transaction into several use cases, as follows. 1) Consumer contacts merchant to encrypt and decrypt security keys, and to request certificates. 2) Payment is transferred from merchant to acquirer (the merchant's bank), and then to issuer through inter-bank network to authenticate the payment information. 3) Acquirer notifies merchant about the status for merchant to fill the order. 4) Merchant requests acquiring bank regarding settlement. 5) Issuer charges the consumer's credit card and notifies cardholder. 6) Consumer pays fee, and acquirer then credits merchant for the exact fee.

The conventional design doesn't have the dynamic control to have customer insight while the issues are resolved in component-based design. This is the service transformation and provisioning example which mediates the enterprise components in forming the skeleton. Service components derived from the business use case are the mediation between business and technical elements which integrate all business semantics, interfaces and technical configurations into a logical component; initially all the service components are at the atomic level [14] [15]. In our component validation and analysis research, we applied a group decision model in the validation process, so the business component with its associated service components are selected and validated through the decision process. We further detailed the rational model in analyzing common components and aggregated services by using a use case parser to determine enterprise components in forming the skeleton of integration. Aggregation is a kind of service normalization in grouping all atomic service elements from fine-grained to coarse-grained components.

We also introduced the component design and interconnectivity for how the components communicate with each other and how the micro process of business semantics is derived in each component. The component element is the physical element that mediates service and information; the design of service components follows our previous strategy, by using business semantic injection to take the context into the semantic and micro processes, so that each component has its contract level responsibility in provisioning a subset of services, and the interface helps the coordination between components and metadata build the foundation of interoperability within the service layer and between layers.

6. Conclusions

Componentization is the first step in service design. As the service concept is moving, global regulation is changing and the high demands in transforming the system silos to an industry mainstream, the financial industry is facing a big challenge within enterprise integration

In order to narrow down the root issue within a small scope as a service component while maintaining its alignment to both business and technical components, an example of a typical issue in a credit card scenario is given in this paper; this service component architecture speeds up the change and dramatically enhances the integrity and responsiveness. An example of service provisioning would be the case where several key industry vendors have built up an industry framework for service providers in that particular industry to follow, starting from business domain analysis and working through to component deployment.

Copyright © 2007 Inderscience Enterprises Ltd.

Any business activity becomes easier when the various parties involved have a common understanding and consensus on business terms, service definitions, interfaces and component specification, and these become transparent and common practices [16]. This objective is built on the foundation of component-based design; we applied both macro and micro process in constructing the foundation, and further aggregates the components into common components and aggregation services. This encompasses component-based design and interconnectivity, transformation of service aggregation through rule-based design, service provisioning in a particular dimension and service virtualization. The important concept this paper derives is that once the metadata of business, service and information constructs an industry standard, the organization's vertical skeleton has the opportunity to integrate across multiple organizations for an easier global practice; this is a long-term industry objective which we call service virtualization, and in this pioneer work we are influencing the financial industry in achieving the objective of component-based integration and service virtualization.

References

Curtis, K., Foster, P. W.and Stentiford, F. (1999) Metadata - The Key to Content Management Services, Proceedings of the Meta-Data '99, The third IEEE Meta-Data conference, April 6-7, 1999, Bethesa, Maryland, USA

Yu, L. (2005) Categorization of Common Coupling in Kernel-Based Software, 43rd ACM Southeast Conference, 2005

Lytras, M. (2006) The Semantic Electronic Government: knowledge management for citizen relationship and new assessment scenarios, Electronic Government, an International Journal 2006 - Vol. 3, No.1 pp. 5 – 17

Dsouza, D. (2001) Model-Driven Architecture and Integration Opportunities and Challenges, Kinetium 2001

Melnik, S. (1999) Generic Interoperability Framework, Distributed and Parallel Databases Journal 1999

Oberle, D., Staab, S., and Eberhart, A (2006) Semantic Management of Distributed Web Applications, IEEE Distributed Systems Online, vol. 7, no. 4, 2006, art. no. 0605-o5001

Staab, M., Stojanovic, N., Studer, R., and Sure, Y. (2001) Semantic Portal - The SEAL approach, to appear in: Creating the Semantic Web, MIT Press

Valkeapää, O. and Hyvönen, E. (2006) Semantic Annotation with Browser-based Annotation Tool, 1st Asian Semantic Web Conference

Barcia, R. and Brent, J. (2005) Building SOA solutions with the Service Component Architecture, IBM Research

Schmalz, M. (2003) EITH – A Unifying Representation for Database Schema and Application Code in Enterprise Knowledge Extraction , 22nd International Conference on Conceptual Modeling

Zimmermann, O., Krogdahl, P., and Gee, C. (2004) Elements of Service-Oriented Analysis and Design, IBM Academy

Gramond, E., and Rodger, S. (1999) Using JFLAP to Interact with Theorems in Automata Theory, Thirtieth SIGCSE Technical Symposium on Computer Science Education, ACM Press 1999 page 336-340.

Hunter , J. (2001) MetaNet - A Metadata Term Thesaurus to Enable Semantic Interoperability Between Metadata Domains, Journal of Digital Information, Volume 1 Issue 8

Copyright © 2007 Inderscience Enterprises Ltd.

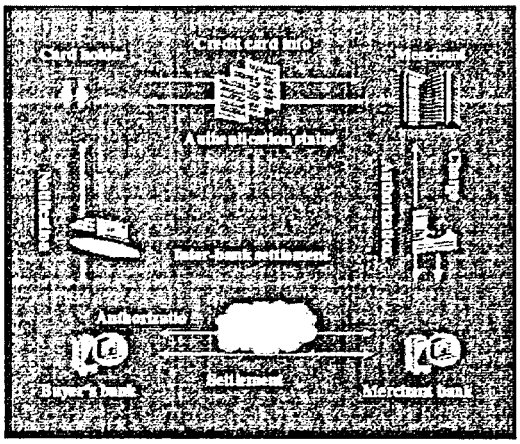


Figure 1. Business use case in the financial industry

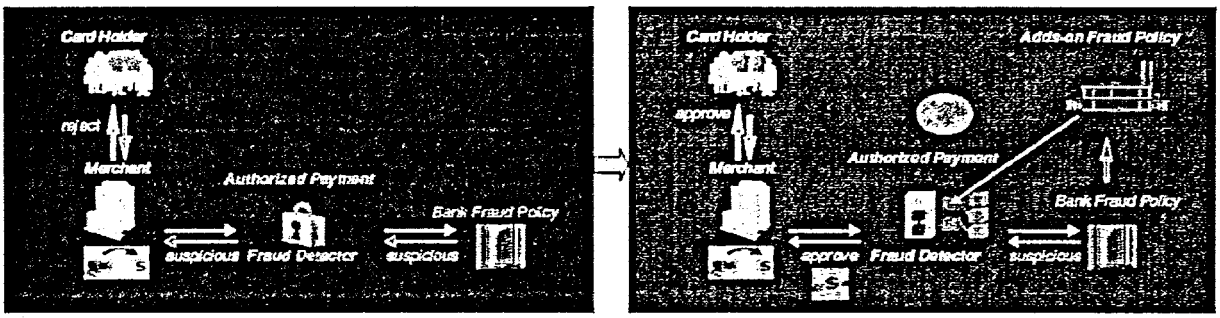


Figure 2. Loosely coupled component add-ons solution for fraud detection

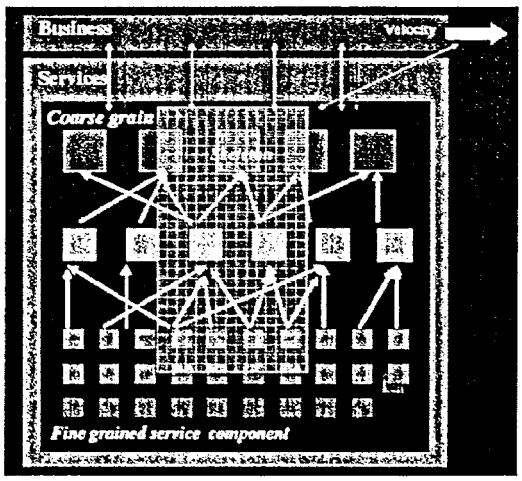


Figure 3. Transformation of common components and service aggregation

S5 Enterprise Interoperability and Service Virtualization

Publication: ACM NZ
ISSN 1176-9998 (http://www.acm.org/chapters/acm_nz/bulletin)

Publication Contact: Taiyu Lin,
Inf. Systems Department, Massey University,
New Zealand t.lin@massey.ac.nz



Date: Fri, 29 Jun 2007 21:49:38 +1200

From: "Taiyu Lin" <taiyu.lin@gmail.com> Add to Address Book Add Mobile
Yahoo! DomainKeys has confirmed that this message was sent by gmail.com

To: "Raymond Wu" <rwu_academic@yahoo.com>

Subject: Re[2]: acm_nz: submissions

Dear Raymond,

I am pleased to announce that your paper titled "Enterprise Integratic Interoperability and Service Virtualization" has been accepted by ACM_NZ Bulletin.

Please be aware that the acceptance is based on the condition that you carefully revise your paper according to reviewers' comments. You will need to document how you treat reviewers' comments in your revised manuscript. I hope that we can receive your revised manuscript soon.

Regards,
Taiyu Lin

Enterprise Integration Interoperability and Service Virtualization

Raymond Wu
IBM Software Group
rwu_academic@yahoo.com

Abstract: *Enterprise interoperability is the ability to exchange and integrate meaningful information through micro and macro synthesis processes, it is supported by component identification, interconnectivity, service state transfer, aggregation, dimension and service main route strategies. The main purpose of enterprise interoperability is to build up a solid foundation for coping with dynamic changes in technology trends, time to market, product innovation, service behavior and enterprise strategy. Service virtualization is a new area which requires attention, as enterprise integration and the industry silo need to be consolidated to form a main route; the service grid concept will provision a transparent and highly integrated framework to provide a global service infrastructure not limited by geography or the system being used.*

Keyword: Enterprise integration, Service virtualization, Interoperability, Synthesis.

Introduction

Service virtualization is the objective in enterprise integration, although interoperability is the intermediate step in achieving this goal. Service virtualization is a realization of enterprise integration that services will be transparent while the alignment between business semantics, services and physical components become visible. It requires a high maturity of both methods and objects across industry to push the common services into mainstream, so the industry will have same terminology, semantics, services and components to serve as foundation of common practices. The fact is that in a chaotic world, this is not achievable from both scale and complexity perspective which need foundation of multi-steps approach, and this is far beyond the capacity of current IT industry. For this reason, this thesis proposes a framework approach to develop interoperability and achieve service virtualization.

The reason this study proposes interoperability as an intermediate solution is that, as it is all aware, a consensus on enterprise interoperability in the vertical direction is in high demand and yet a confused design has resulted in businesses losing their enterprise governance in order to accommodate visibility in semantics, syntax and architecture alignment. In other words, enterprise has been unable to effectively leverage business, service and technology into a common vision. Therefore, to achieve service virtualization, the enterprise needs to aim for interoperability and first build a framework and strategy, such that the integration is not limited to layer-to-layer or across layers, but instead follows the nature of business to implement all semantic metadata in a component-based foundation for both micro and macro designs.

First, this paper would like to clarify the terminology of interoperability and virtualization this study uses. Interoperability is the ability to exchange and integrate meaningful information [1], while enterprise interoperability is the key concept for integrating various kinds of information stored in a variety of repositories [2]. In practice, however,

interoperability is absent in enterprise integration due to incompatible software, inadequate data formats, semantic heterogeneity and data models [3]. The difference between integration and interoperability is that integration deals with a synthesis of information from disparate sources on a conceptual level, while interoperability deals with making it possible for information from disparate sources to be aggregated operationally for use by the system. This paper wish to highlight that integration is strategic synthesis while interoperability deals with an "aggregated" semantic. Therefore, interoperability is not a necessary condition for conceptual integration though it is reasonable to assume that interoperability facilitates such integration, which relies on the metadata interoperability in achieving the goal.

The importance of enterprise interoperability is illustrated by the following example. An "account balance query" may consist of a combination of "multi-account identified" functional components and "database query" non-functional components. The service aggregation parsing process identifies these two service components as candidates for common service; they are validated once the frequency of the combination passes the appropriate threshold and their interconnectivity design is seen as feasible. Service metadata holds the mapping and routing intelligence that is stored in a service repository so that both common components and the service aggregation have an opportunity to form the main route of the common service skeleton. Main route stands for the highway and industry common practice constructed by integrated common services. Any business semantic translation to the process and infrastructure level becomes feasible and efficient under this skeleton [4].

In taking this concept through to realization, service virtualization requires industry to follow a common framework as a foundation of common services and components, so that consumers of the service share the same protocol, interface, services, components and information in a transparent environment. Enterprise interoperability is a crucial checkpoint along the path to service virtualization, because in the real world, industry has many different methodologies, systems, terminologies, varieties of architecture style and integration strategies. This research need a structure with flexibility in implementation of details, however this study must also create a framework which allows visibility of how the strategy and roadmap impact the transformation. For this reason, this study proposes a three-step approach consisting of ontology, framework and virtualization. The interoperability of these three steps is shown in Figure 1, and then described.

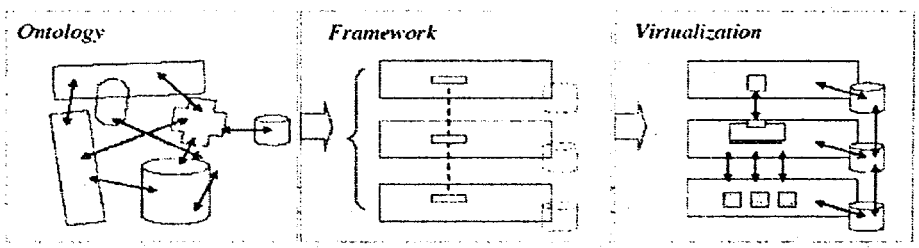


Figure 1. Three steps in interoperability approach

1) Ontology – This is the initial step and the preparation for the framework. This step aims to collect all methodology, systems, terminology, architecture styles, interference and

feasibility studies for consolidation into a common view to act as a main route for the enterprise transformation. This step also involves a conceptual strategy mainly focused on the foundation of component analysis, validation and semantic strategy. Semantic strategy stands for the methods for maintaining the semantics coherence when they are deconstructed from business component into atomic elements. So before the enterprise moves on to the framework stage, all the heterogeneous concepts and strategies have the opportunity to be consolidated into a homogeneous solution. Simply put, the ontology step gathers all related technologies and solutions within the enterprise, and re-creates visibility from the view of interoperability. From the organization's perspective, as many IT professionals are still confused about the strategy and terminology of interoperability, enterprise ontology should, through a multi-disciplined approach, be able to sufficiently educate people to create a common view in transforming the strategy and plan into the framework stage.

2) Framework – A framework is needed to encompass strategy and lifecycle roadmap as a critical milestone to achieving enterprise interoperability. The reason for this is that when the enterprise pushes the implementation silo into the main route, it requires a multi-dimensional mechanism to encompass interoperability, at both micro and macro levels, to retrieve the assets from the ontology-stage outline into synthesis process which mainly covers aggregation and common components refining. The philosophy of the framework proposed in this study is that enterprise interoperability should not be approached on an ad-hoc basis; key issues will never be resolved unless semantics (business) and metadata (techniques) have the opportunity to be initially synthesized in a micro process which is the atomic component level. This vision, which is the foundation of the proposed framework, is still new in this field. The concept of this study is that study should not focus on interoperability at merely the system or application level. Instead, in this new vision this paper will be using such terms as components, services, interoperability, semantics and virtualization in a multi-dimensional integration approach.

3) Virtualization – This is the final step of integration and tuning before moving into industry common practices; to achieve service virtualization it is essential to minimize the gap between the proposed strategy and industry. Service virtualization is encountering high demand. For instance, the Zachmann enterprise framework and eTom in the telecom industry broadly function as industry foundations; however, they have been insufficient to many scenarios such as joint-venture, acquire and merge cases. The main reason is that the industry main route is still far from common practices because the industry semantics standards are still at a macro process level instead of a micro element breakdown; as a matter of fact, many integrations are still in a case-by-case or n:n situation and lack a robust driving force working as a single source of truth in both method and semantics to drive the transformation into an n:1 evolution. This study consider the root issue to be weakness in the semantics synthesis process; in other words, the mediation of services has not been enriched. An enriched service layer means synthesis of components and alignment of context, syntax and metadata have reached a mature stage, so semantics can freely transmit across the service layer and expand to industry scale. For this reason, the enterprise framework is sufficient for interoperability yet insufficient for service virtualization due to the weakness in stretching out to the industry main stream. Therefore, further investigation and approaches are proposed in this study, which intends for service mediation, industry framework and event-based technology to move enterprise interoperability into industry common practices. The concept of

interoperability has already been implemented in several projects with convincing results.

Literature Review

Enterprise integration and the use of common components enable leading companies to integrate their business and decision support systems with customer relationship management and supply chain management systems. Interoperability supports the collaboration of business partners as they produce value and services in the manufacturing, supply chain, distribution and order management lifecycle [5]. As these practices continue widespread adoption, corporations will expect the same level of collaboration and support from industry for common practices. Enterprise integration is fundamental to the solutions pursued by architects to meet the objectives laid out in 2002. On February 6, 2002, the Office of Management and Budget began a significant effort aimed at identifying opportunities to simplify and unify processes, and to support industry through effective integration to provide a framework of universal Internet-based specifications and standards to support the secure sharing of data and business processes among agencies. The reference model is shown in Figure 2, J. Qin and J. Prado, in their "The Semantic and Syntactic Model of Metadata," further explain the ontology-based approach for taking both semantics and syntax into modular model metadata, particularly in light of gaps between metadata semantics and syntactic structures resulting in duplicate efforts in binding the same data elements with various languages and application programs. This has led to widely varied data binding models and implementations, while common semantic elements in metadata schemas such as title, author, subject index terms and description are often far from enough when finer metadata representation is needed [6] [7].

From architecture style perspective, Service-Oriented Architecture (SOA) is commonly launched for Enterprise Integration and service base implementation. Semantic annotation of all aspects of web service, including capabilities, data and processes, is an important step towards making SOA a success. In 2006 C. Pedrinaci proposed the Event-Driven Architecture (EDA) coupling with SOA to obtain a particularly appealing architecture for supporting the effective execution of possibly distributed and completely decoupled services, synchronously or asynchronously executed in an efficient manner. Designing new systems will require establishing an agreement on which events there need to be, their syntax and their actual semantics. Thus, adapting—or, as this study prefer to call it—mediating between data models remains a critical requirement.

Service Interoperability

In the previous terminology, those components with a high frequency of utilization are termed common components. At an enterprise level exceptions in handling a particular requirement are normally not allowed, in order to maintain high integrity within the enterprise; therefore, consensus and enterprise standards are usually required at the outset prior to the design. As aggregated services and common components are critical in forming the enterprise skeleton, the design and aggregation process must be cautious from an interconnectivity perspective.

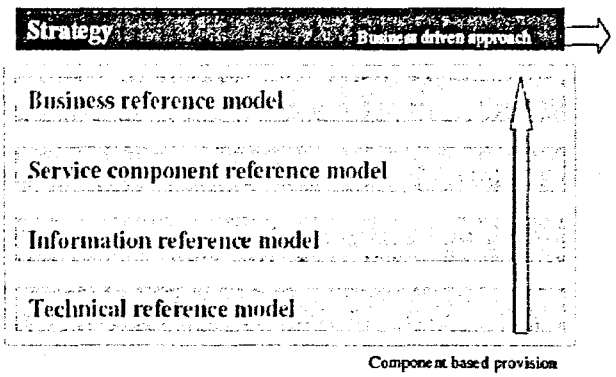


Figure 2. Steps in interoperability approach

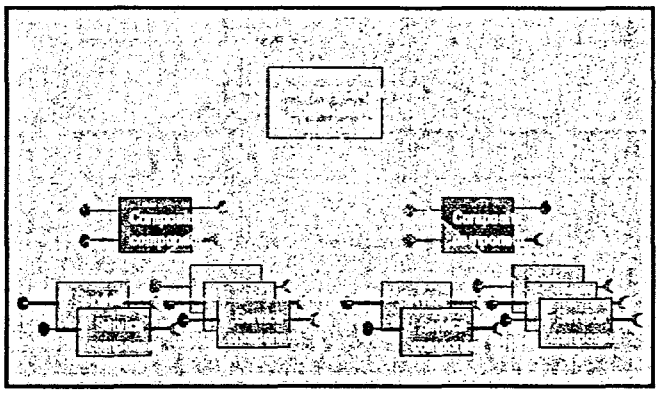


Figure 3. Service aggregation and synthesis

Figure 3 is being used to present the concept of service aggregation and fine-coarse grain synthesis. This implies that high integrity, optimization, grain size, dimension in service provisioning, visibility, traceability, performance and other factors are areas which must be brought into consideration. Once all of these skeleton elements are well specified and designed, the "main route" needs to be outlined at a state-of-the-art level. What this study term the main route is based on the top of the enterprise skeleton this study build; the value-added service may mediate this aggregation and form a long-running process as a skeleton for a business line. One example is in the credit card business in which aggregated services have been built between the card holder and card issuer; in this case the same aggregation is also being formed between the merchant and the bank. The mediation service of an inter-banking fraud detector component has been designed by reading fraud profiles in real-time transaction environments to mediate the individual services into a business main route; this highly enhances the service provisioning in a particular dimension.

The design of service components follows previous strategy of this study by using business semantic injection to take the context into the semantic and micro processes; this means each component has its contract-level responsibility in provisioning a subset of service. This interface helps the coordination between components and metadata to build the foundation of interoperability within the service layer and between layers. In the following paragraph this paper take the results from component design into the service level while applying automata theory to build up the rule-based service aggregation at the enterprise level. As this study is still at a pioneer stage in applying automata theory in service provisioning and aggregation, and this cannot be accomplished within a short period. Further research efforts will be undertaken following the current work [8].

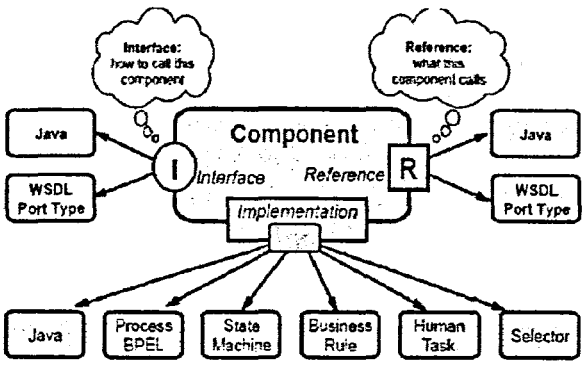


Figure 4. SCA main composition

SCA (service component architecture), as shown in Figure 4, is the framework for micro process design of service components; SCA deals with component design and interconnectivity in micro mechanisms [9]. The service interface specifies SP (service provider) implementation and is invoked by the SC (service coordinator). The SC has the capability to invoke SPs through the service bus without needing to know what SPs are and how the interface was configured. SCA deals with component design and interconnectivity in micro mechanisms and consists of three main areas:

1. Interface: Luca et al. demonstrated their surveying results from the theory behind formalizing rich interfaces for software and hardware components. Rich interfaces specify the protocol aspects of component interaction. Their formalization, called interface automata, permits the compiler to check the compatibility of component interaction protocols. Interface automata support incremental design and independent capability in implementation. Incremental design means that the compatibility checking of interfaces can proceed for partial system descriptions without knowledge of the interfaces of all components. Independent capability in implementation means that compatible interfaces can be refined separately while still maintaining compatibility. This study used both of these methodologies in component interface design [10].

2. Implementation: Service components are the main body of SCA, and include: a) Java objects, normally working in a mediation module either in WSDL (web service definition

language) or java interface type; b) business process, a combination service made up of service components with a specific order to achieve a business goal; c) variables, storing the data used by a business process; d) microflow, a process that is contained within a single transaction; e) long running process, a process which is executed over an extended period of time and is much more flexible and resilient than a microflow; f) human task, a unit of work done by a human which involves the interaction with other services; g) state machine, an event-driven business transaction in which external operations trigger changes that guide the transaction from one discrete mode to another; h) state, one of several discrete individual stages that are organized in sequence to compose a business transaction; i) composite state, an aggregate of one or more states; and j) business rule, which can be anything that captures and implements business policies and practices.

3. Metadata: The parameters transmitted in and out from a service are modeled as service data objects implemented by Java class. SDO (service data object) complements SCA by providing a common way to access many different kinds of data. The specification reduces the skill level and the time required to access and manipulate business data. Today, a great many APIs are used to manipulate data. These APIs tend to tightly couple the source and target of the data; this makes their usage error-prone and subject to breaking as business requirements evolve; however, SDO makes it easier to use and thus realize the value of these APIs without having to code directly to them.

SDO is designed to unify data for applications to access heterogeneous environments in a uniform way. So all industry key data formats such as RDBS, XML, web services and metadata can be unified based on this approach. It was introduced by BEA and IBM, and now is widely used in key vendors of the industry. SDO is based on the concept called disconnected data graphs. A data graph is a collection of graph-structured data objects which allows the client to retrieve, mutate and change the data graphs out and in from the data source, and this is supported by the data mediator services which handle the process between applications and data objects. Therefore the fundamental SDO architecture consists of the front-tier data mediator services, the process-tier data graph and data objects in the resource tier, which enable both static and dynamic (loosely typed) data mediation. It also provides a metadata API, which allows applications, tools and frameworks to introspect the data model for a data graph.

When SOA is used in achieving interoperability together with semantics, the question may arise of how to maintain loose coupling, yet maintain semantic integrity. How to tell if a service is similar to this study's requirements, or do services just "hide" APIs? To optimize different scenarios for an elegant solution, SDO is targeted in static type to bypass API, and target event-based technology and aggregated services for dynamic type to fill in the gap between service orientation and semantic integrity, as SOA needs to be coupled with EDA to cope with semantic integrity. In summary, for those loosely-coupled long processes, this study propose the following two approaches:

a) Event-Driven Architecture coupled with SOA was indicated in the literature review for supporting the decoupled services in an efficient manner. Event-based communication is indeed useful for a decoupled integration but it does not, and it actually cannot, avoid the need to agree on the events and the need for mediating between heterogeneous data models

which must co-work with SDO.

b) Dimensional service is a new concept supported by the service orientation approach. The concept requires maturity in service orientation to provision patterns of "stabilized service" as coarse-grain service, to eliminate the boundary of semantic transmission, which can eliminate dynamic semantic mediation. This will be discussed in the next section.

Semantic Mediation

This study further proposes the service dimension concept as follows:

- The first dimension is component-based implementation between human and machine through deconstruction process. One of the example is, the bankers who issues credit card identifies S&M (sales and marketing) and CRM (customer care) as business components to launch "frequent flyer points" program (FFP) for service promotion in airline ticketing. The business components are therefore, decomposed into groups of elements, the elements will be further identified to form vertical alignment.

- The second dimension is semantic transmission through metadata design and synthesis of micro and macro processes. By taking the above example of credit card issuer into FFP semantics transmission. The implementation takes approaches of semantic mediation across common component between card transaction and ticketing. So the significant accounts (VIP) can be identified for common services between agents and issuers.

- The third dimension is semantic implementation under enterprise vision, so once the three levels of semantic mediation are implemented, there is opportunity for atomic components to be designed in a synthesis which consists of micro and macro processes. Component synthesis between card issuer and travel agents will form a coarse grain service for semantic implementation between "ticketing" and "card transaction", in addition metadata hold the alignment coherence in fine-coarse grain services implementation, so a mutual benefit can be expected through transaction base analysis and consumer's insight.

From the perspective of industry standards, semantic mediation as proposed in this study has been expected to produce significant benefits to industry. Following are some typical scenarios.

✧ Large-scale global-local topology, in which the enterprise architecture requires a semantic metadata-centric solution to maintain the coherence of semantics across organizations in synchronization with the context of corporate strategy.

✧ The interference between business and IT is significant, such that the company's KPI (key performance index), service and infrastructure requires effective semantics to transmit across the vertical direction and the ontology of the alignment requires identification of common components and a method of rapid implementation for those new alignments of semantics, and service virtualization in the industry's main route becomes a high demand.

Service dimension is semantic implementation under enterprise vision, so the coherence of

semantics can be achieved and services can be reused in the whole enterprise. In addition, once the three levels of semantic mediation are implemented, there is opportunity for atomic components to be designed in a synthesis way which consists of micro and macro processes. Meanwhile, as the visibility of vertical alignment is clear between architecture layers, the semantic centric-metadata forms a clear picture of the enterprise skeleton stretched across the interoperability between business, service and information. Through a semantic mediation approach, this study can either reuse the metadata mapping and configuration from that successful experience or construct a new map for the new requirements by using the same methodology. Consequently, once the service components and aggregation services are built they can either be reused or aggregated to a coarser grain if the aggregation is common and optimized in the parsing evaluation.

Service Virtualization

Aggregated services and common components should not be tightly coupled; instead, their dependency and interconnectivity should be based at a parameter and metadata level and their design and interfacing specifications should be fully visible for future development. The reason for this is that the more significant role these skeleton elements play, the higher the complexity of interconnectivity, business criticality and access frequency, which can result in significant impact if a defect is detected too late. In many instances, a defect is a kind of binding between components instead of a coordinate from aggregation. A service aggregation needs to be implemented in an empirical rule-base in the same way that a phrase needs words, grammar, semantics and a certain level of common recognition. It should not be allowed to be handled for individual needs differently or randomly; this is the reason it should be designed and maintained at the enterprise level as a single view of enterprise solution.

The micro process starts with the mechanism for how the component is designed; this involves taking the business semantic of the macro level with coherence of enterprise interoperability between each layer. Before the design of the service component, this paper take direction from the business component by using a top-down approach to derive the component design while at the same time streamlining the alignment between service components and physical components by using both top-down and bottom-up approaches. Most of the bottom-up approaches are a componentization process working from existing information components, business objects, codes, system metadata or database access protocols.

Once this study construct the alignment between business, service and information and have the service component design and interconnectivity completed, this study actively work on service layer transformation. The reason this step is crucial is that the goal of vertical enterprise integration is to create interoperability between layers from both micro and macro perspectives, with both business and information layers working as a foundation to support the formation of the service layer which provides the actual services. Once the service component has completed its design with alignment to business and information, the focus can be shifted to micro interference within the macro view.

In the previous research of this study on metadata interoperability this study revealed, in relation to both political and rational models, how the components are derived through the validation process and how the strategy of component implementation achieves layer interoperability. This paper extends research in component design and technical implementation for both internal and interface component factors [11]. Figure 5 shows how common services are built on top of the service bus and how common components, information and services interoperate.

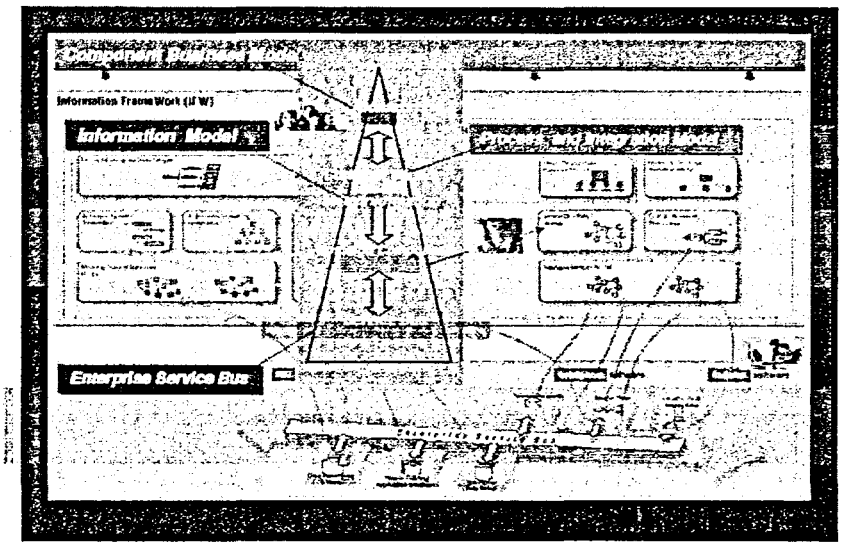


Figure 5. Service virtualization

Conclusion

As mentioned, service virtualization is the final goal and realization of enterprise integration that requires a high maturity of both methods and objects across industry to push the common services into mainstream. In order to encompass the realization in this chaotic world, this thesis proposes a framework approach to develop interoperability and achieve the goal.

This framework, constructed by strategy and lifecycle roadmap, aims at service interoperability as the intermediate step to achieve the goal, such that the integration is not limited to point-to-point, but instead applies semantic metadata in both micro and macro designs and is built on top of the component-based foundation. By taking the approach of industry framework, this thesis proposes industry to follow common practice, so the enterprise can build up interoperability and the service provider can provision its service to service consumers by sharing same protocol, interface, service components and information in a transparent environment. The integration uses a vertical interoperability approach within the enterprise and a horizontal approach across enterprise and industry.

Enterprise interoperability is built on the foundation of a validation process of decision, design and provisioning stages which consolidates many different methodologies, systems and terminologies, and a variety of styles of architecture and integration. It uses both micro and macro approaches in the vertical direction; there are three steps in achieving the goal: Initially, an ontology needs to gather all heterogeneous designs to work on their interference and consolidation into the main route. Second, a framework and roadmap are needed to build the foundation of interoperability; a multi-dimensional mechanism needs to be implemented to handle interoperability for both micro and macro processes to integrate the assets within the ontology. Third, the integration and tuning between theoretical research and industry common practices, to minimize the gap between the strategy of this paper and industry, is the realization step in achieving service virtualization.

SDO, semantic mediation, EDA and service dimensions are all used to achieve service interoperability as they fill in the gap between component-based design and interoperability requirements, and support service orientation such as SOA and web services as a supplemental solution in streamlining loose coupling in a tightly binding environment. In enterprise service provisioning and virtualization, this study constructed the skeleton of enterprise service with leverage to each layer and interoperability of the layers. Service virtualization is a process of leveraging enterprise integration strategy within a specific geographical region or across regions. This can be an industry standard scenario such as a credit card transaction in which the service, geography and systems are fully transparent to both cardholders and merchants. A full virtualization environment needs to be built on top of the foundation of component design and service transformation process in order to apply the vertical enterprise integration approach to this final stage, which is the goal of this research work.

References

1. Vckovski A., Brassel K. and Schek H., (1999). Interoperating Geographic Information Systems, Second International Conference. INTEROP March 1999 Computer Science 1580 Springer
2. Hadzilakos Thanassis, H. G., Kavouras Marinos, Kokla Margarita, (2000). Interoperability and Definition of a National Standard for Geospatial Data. International Journal of Applied Earth Observation and Geoinformation Vol. 2(Issue 2): 120-128.
3. Vckovski A., Brassel K. and Schek H. (1999), Interoperating Geographic Information Systems, Second International Conference, INTEROP March 1999 Computer Science 1580 Springer
4. Curtis, K., Foster, P. W., Stentiford, F., (1999), Metadata - The Key to Content Management Services, Proceedings of the Meta-Data '99, The third IEEE Meta-Data conference, April 6-7, 1999, Maryland, USA
5. W. O'Brien, R. Issa, Z. Shen, and H.Xie. (2004), Configurable Environments: A Vision for Future Project Information Technologies. In *Knowledge Management in the Construction Industry: A Socio-technical Perspective*, S. Kazi, editor. 2004
6. Qin, J. and Prado, J.C., (2006), The semantic and syntactic model of metadata, Série Comunicação da Informacao Digital 2006 Vol. 4 (pp. 143-156)
7. Qin, J. and N. Hernandez., (2004), Ontological representation of learning objects: building interoperable vocabulary and structures, ACM Press New York, May 17-22, 2004, 348-349
8. R. Barcia and J. Brent, (2005), Building SOA solutions with the Service Component Architecture, IBM Research
9. Wu, R., (2007), Semantic Metadata in Enterprise Integration, IEEE Southeast 2007 conference, pp 233-236
10. L. de Alfaro and T. A. Henzinger, (2001). Interface theories for component-based design, in *Proceedings of theFirst International Workshop on Embedded Software* (EMSOFT), Lecture Notes in Computer Science 2211,Springer-Verlag, 2001, pp. 148-165.
11. Jie Lu, Guangquan Zhang & Fengjie Wu, (2005), Web-based Multi-Criteria Group Decision Support System with Linguistic Term Processing Function, IEEE Intelligent Informatics Bulletin June 2005 Vol.5 No.1

IBM

DCRP Strategy in SOA realization

Distribution: IBM - ASEAN Software Group SWITA
Version 1.0
May 8, 2006
Version Number: 1.0
Date: 05/08/2006
Author: Raymond Wu, Consulting IT Architect

Summary

This paper conducts a research starting from business semantic input, enterprise strategy to business processes, to translate and coordinate business goal into IT development and realization of SOA, this imply the enterprise architecture need a robust mechanism of semantic metadata to coordinate each repositories, so the enterprise strategy impact the business process effectively, and streamline the succeeding services and information components development. There are two processes to achieve this objective.

The first process validate business semantic and enterprise goal into business process and its service components level which we call "business semantic injection". This process is an innovation in enterprise leverage to service oriented concept of "separation of concerns" of business component which is above business process level. The key of this process is componentization through decompose process which we identify the business component and its associate service with details specification of the business semantic, choreographic interfacing, database and java referential and status in semantic and service meta repository.

The second process is beneath the business process level which we call "service metadata repository" which take the business semantic forward and register atomic service component into service repository, so semantics and metadata of their collaborative choreograph into repository. The key of this process is aggregation through compose process, as the atomic service is well-defined for component internal, interfacing and coordination, we can take the output from decompose process and aggregate into composite service which is common in the business services.

DCRP taking SOA, meta repository and business semantic concept, in building up an enterprise architecture foundation of high responsiveness and reuse environment. This was the main purpose of my research motivation.

Background

IT industry became challenging in 21st century for its significant growth of enterprise level IT systems to handle huge volume transactions and sophisticated businesses.

In this new computing age of large scale, high complexity and dynamic development, a common weakness in the interoperability between business and IT results in poor business responsiveness and IT governance to achieve the enterprise goal.

Enterprise architecture mainly mediate business and IT and minimize the gap by improving the governance, agility and business integrity in a rapid development environment. However the realization of enterprise architecture goal is still far from mature stage to most of the organization including over 90% of medium to large scale of heavy IT users groups.

In the recent 5 years, we had SOA, MDA, EDA and many leading IT methodology or style of architecture try to solve the problem. These resolutions already bring to us significant improvement as they enhanced the business decomposition and modeling, and created loosely coupled, reuse and agility in IT responsiveness to achieve business goal and process re-engineering, however these are still not enough to cope with the new decade challenge particularly for those large scale and high complexity enterprises for the next 10 years. Following diagram show the evolution stages.

The enterprise architecture has its grey area need be resolved. The investigation shows; the architecture solutions mainly work on the integration inside one horizontal layer (such as modeling inside business layer, decomposition in process layer and API/XML in data layer) but lack of a robust mechanism to translate semantic metadata vertically to coordinate enough information between layers and streamline the interoperability between business and IT. In the other words, the integration works effectively between same horizontal layers but not cohesively between vertical layers for semantic transformation, this supplementary solutions built on the existing foundation and "ontology metadata" is our strategic solution to cover this grey area.

In SOA Process Decomposition, all of the processes are decomposed into sub-process and use case to derive service element and component.

The key is "decomposition" is a way of "separation of concern" to derive service, the other purpose is taking the business modeling output and adding the business information to the fine grained element to perform "business centric" modeling.

What will be the decomposition strategy in order to take output from BSM and work effectively for ontology metadata objective, this require "atomic" level service produced from decomposition process to take the "business semantic injection" coordinated to business metadata in to add business semantic into information metadata.

This imply the service component need to hold information for both business metadata and information metadata. Fine grained service component has capability to aggregate the associated component into coarse grained service, however this aggregation process need be bottom up once completing business injection process it's impossible for the coarse grain to absorb a larger scope of business semantic without "divide and conquer" and assembly from atomic level service aggregation.

SOA enhanced IT responsiveness to take the business decomposition output and streamline services and its interoperability across the different areas *(step 1 in diag. 2)* ontology metadata map the enterprise semantic such as goal and KPI to business semantic and create a common foundation of "semantic injection" in service component level *(step 2 in diag. 2)* so each service component holding related business semantic and rule, plus the information inherited through ontology metadata mapping. These 2 micro steps are what we call supplemental "vertical" process on top of SOA process decomposition.
The main purpose of Ontology enterprise metadata - *OEM* is related to the injection process of semantic

information into micro components initiated by 3 mapping steps;

a) A top down matrix mapping from enterprise strategy to business metadata, the mapping information is stored in repository.
b) A bottom up data aggregation and system abstraction from infrastructure
c) Using the method of "early exposure of enriched business metadata" stored in OEM repository to coordinate the top down and bottom up metadata as a foundation of OEM big picture.

Root issues study and research in resolution

1 **Misalignment of IT to corporate strategy**
The lack of a clearly communicated business strategy creates an environment in which efforts are tactically focused on immediate business needs without regard for the future long-term direction of the company Collaborative design and enterprise view is required in development environment rationalization

2 **Lack of effective business or IT governance**
Without effective business and IT governance, lower-value service requests receive the same priority attention and resource allocation that higher value service requests receive System implementation need an overview from corporate level to prevent isolation and non-standard implementation

3 **Proliferation of applications**
Without the necessary controls to ensure that each technical investment is justified in terms of business value and adheres to the technical standards, Client application portfolio will continue to grow larger, more complex and more expensive Keep a close look at ROI so each investment need be business feasible

4 **Duplication of functionality, data systems and support infrastructure**
Many times individual business units have the ability to select the technical means they feel best support their needs. This unfortunately creates a situation where redundant capabilities, data and infrastructure are purchased, duplicating support needs and cost

5 **Inappropriate enhancement and use of systems**
Users have focused their technical resources on tactical system enhancements that have resulted in modifications that meet immediate needs, often at the expense of the longer term direction or value. A view of priority and roadmap need be provided at overall enterprise level to prevent resource misuse or mis-focus

6 **Silo Operations**
Too often individual business units an organization operate in a very autonomous, silo manner, limiting the leveragability of technical solutions, data and infrastructure. The result of this situation is a complex support needs and redundant cost structure. Create visibility & registry for each technical implementation, so a server, infra. component need be validated for integration at enterprise level

7 **Fragmented information architecture**
Too often the acquisition, storage, manipulation, and use of data is done disparately across the organization, resulting in data inconsistencies, redundant data purchases, duplicated support structures, and excess cost. Enterprise view and sharing infrastructure is required

8 **Standardize file transfer & messaging**
These is a best practice for layers' communication and message
Transfer, XML is the recommendation for messaging between presentation layer and middleware, and between middleware and application layer.
There is a well-developed middleware gateway between mainframe and unix (presentation web server), the TCP/IP is recommended for communication between common gateway and mainframe, and C:D

(connect direct is the option for file transfer between mainframe and unix server)

9. **Governance of project architecture in high complexity development**
In the following example, team leaders (Data, System, BA, SA, Dev, Conversion, Testing) form a virtual team and communication in project governance, the mechanism of the process route is to predict the defect in a collaboration design. The strategy of this project architecture governance is try to create the route and communication channel for technical leader (from BA to development) to log their concern and change request, so all the system and data experts can work on the business driven issue and provide system level provisioning to support the application, to form a healthy information architecture.

10. **Re-engineer the well-defined interface**
Once we have a well-defined common interface for a common architecture, we can eliminate those ambiguous scope and client site transformation by using this common interface, so client is responsible for taking their existing business component, data, metadata and process functions and convert into this common interface, this minimize the risk and effort, what we need to do is minimize the scope need customization to be less than 30% (ideally 0%). The first step of this concept is creating a common interface for generic purpose. Working on the format and schema of this interface need a survey of industry common requirements and existing architecture, once this is done, we need to work on the process that will take the interface to existing architecture.

11. **Foundation rationalization**
Use existing architecture components, portfolio, methodology, business element and external commercial (common component) to build up the coarse-grained technology foundation as the candidacy of baseline foundation, SOA can be applied to orchestrate and reuse such component to fulfill the services. Most of the effort should be spent on the gap between "baseline" and "marginal" , between marginal to "best practice" is adjustable based on the optimization from case to case

DCRP process resolution leverage to SOA realization

The Reuse methodology is entitled "Decompose-Compose-Register-Provision" (DCRP) is intended to address the labor-intensive maintenance concerns inherent in multiple instances of

Decompose - The functional specifications (business rules) are used to extract multiple "atomic business processes" from the Use Case.

Compose - The "atomic business processes" are re-engineered to allow for reuse, their input and output requirements (i.e., what is passed to them and what is returned from them) are defined and documented, and their business purpose and technical requirements are captured in a searchable database. These reusable components are then crafted into a program to satisfy specific business requirements.

Register - Reusable components are tracked within a reuse register. This register identifies every instance of employment of the reusable component (e.g., every program which contains the component).

Provision - Reusable components are available to provision (provide) their service to any other (reusable-component-based) program which might require their functionality or business purpose.

As in Java, the method grows more powerful over time as more and more components are built and become available for reuse. DCRP is primarily intended for the construction of computer programs. The mere fact that DCRP brokers reusable components need be written only once and then reused in any and all programs .

Similarly, any time a change of requirements is needed in a reusable component, it need be made only once (to the component, itself) and all of the programs that use the component can be quickly identified (through the Register) and recompiled to obtain programs containing the new requirements. The new requirements may necessitate changes to other components (e.g., changes to requisite data strings to accommodate new data fields), but these, too, would be identified through the Register and applied in a uniform (standard) manner.

DCRP pre-build architecture foundation tried to enables, and accelerates the delivery of portfolio offerings

through increased quality, innovation, and productivity. It enables the rapid formulation of new portfolio offerings and their subsequent adoption in the field by supplying much of the application infrastructure. Functionality provided by the Agile Application Architecture is pre-integrated and pre-tested so that solution developers have a head start in development of new applications.

The first step is creating a common interface for generic purpose. Working on the format and schema of this interface need a survey of requirements and existing architecture, 2nd step is we need to work on the process by using the common interface to take existing architecture to the future architecture from data perspective

The DCRP concept of enterprise agility is based on the idea of using standardized components that can be reused and applied across different clients and applications, with minimal effort required to fit them together. The best way to minimize the risk and fully satisfy the client's requirements is using a interface to bridge to transform client's data and functions and clearly define the responsibility for what need to do.

DCRP applications services support organizations plan, develop, integrate and manage custom applications, packaged software and industry-specific solutions. As a result, clients are able to leverage vast industry and application expertise to better support their continuing business objectives, improve operational performance and realize a higher rate of ROI (return on their application investments) while managing cost.

The computing of decompose and compose processes follow SOA principle in building up "separation of concern" and "reuse" environment by using meta repository transformation, and use interoperability and choreographic meta repositories in provisioning the services.

The SOA concept has been common in the industry for several years however, the realization is still far from real world, the business motivation of my research and innovation of DCRP process was mainly focused on high complexity and heterogeneous projects in providing guidelines and architecture framework, it has been applied in some of my previous projects and need some more work on the first process (business semantic).

Principles

The principles listed below are key guidelines for the design or purchase of applications and application components supporting distributed, client/server computing across the state.

Principle 1
Design applications to be highly granular and loosely coupled

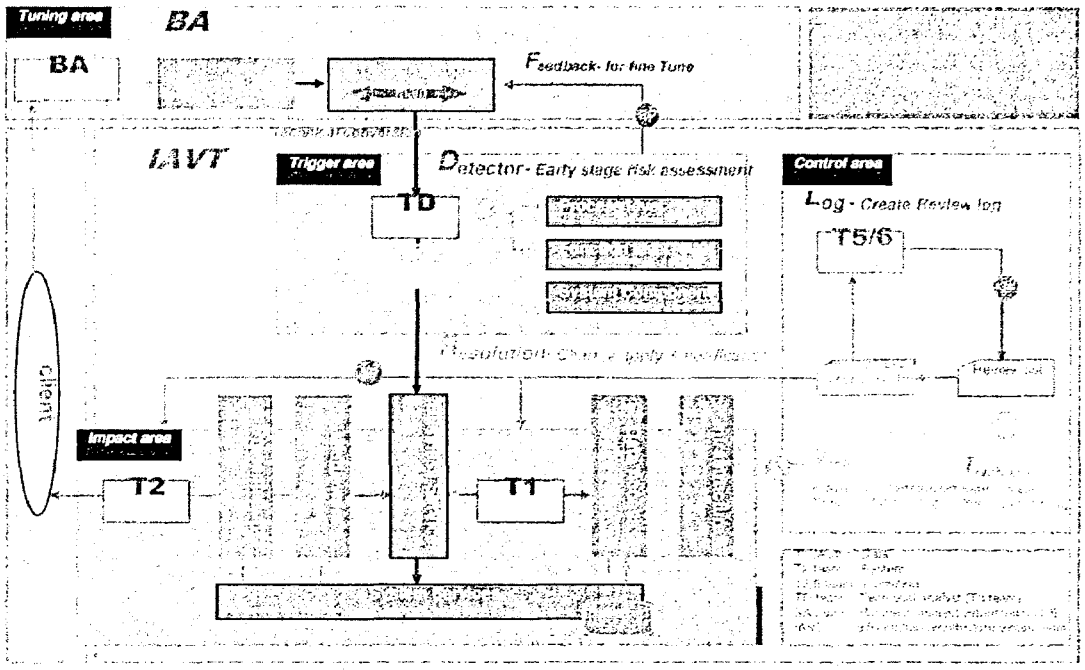
- The designer should allow for the possibility of re-partitioning an application in the future.
- Being highly granular and loosely coupled provide flexibility in physical implementation (i.e., in the deployment of application components on different platforms).
- Highly granular, reusable application components are key to increased productivity and rapid application deployment. The Componentware Architecture supports a highly granular design.

Principle 2
Plan for extensibility and scalability

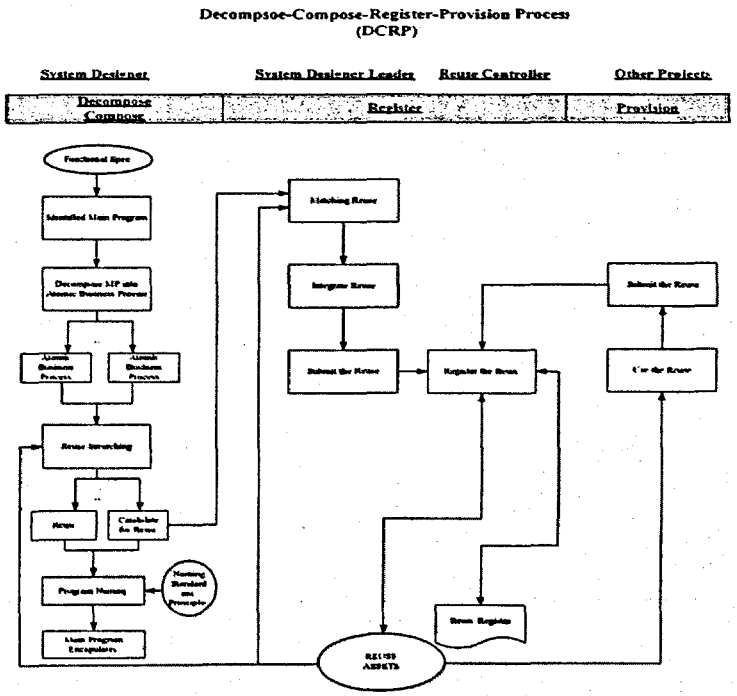
- Applications must evolve to support new business requirements and make use of new technologies.
- Extensibility provides functional scalability.

Principle 3
Design application to reuse components

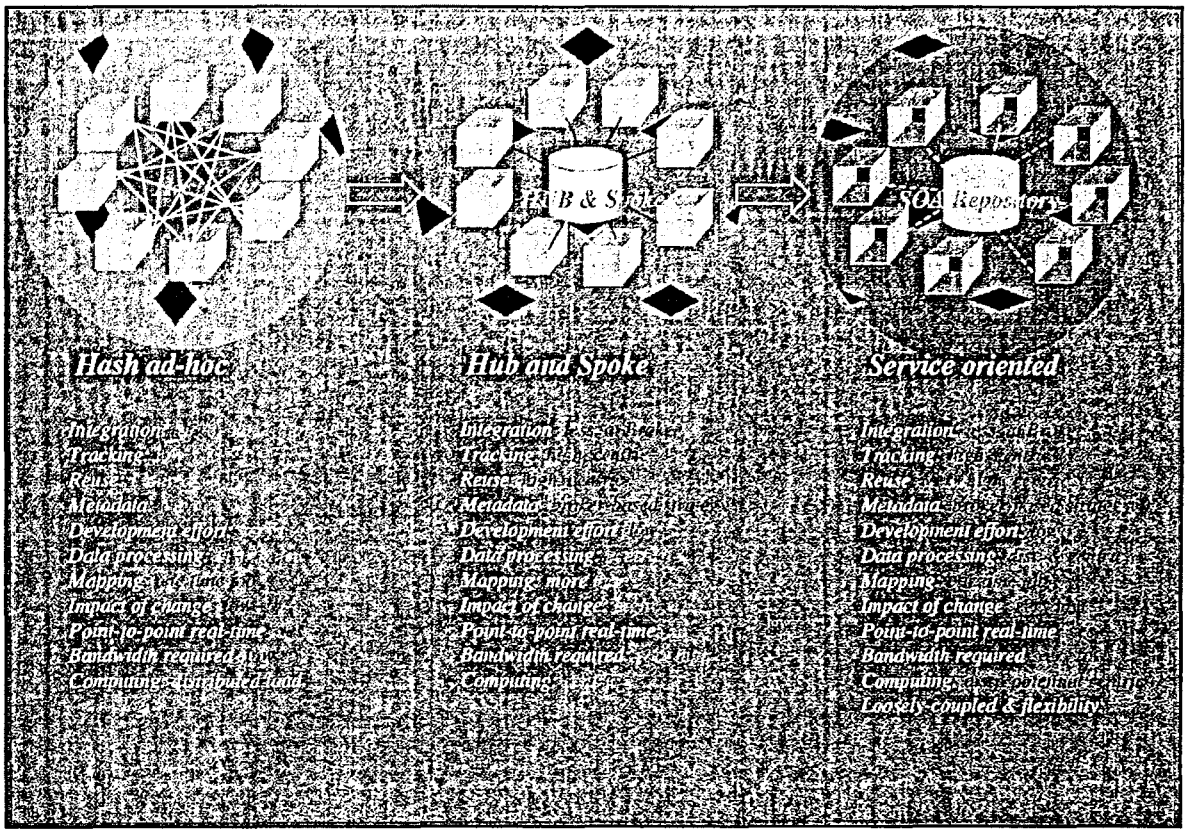
- Applications should be built by assembling and integrating existing components, rather than

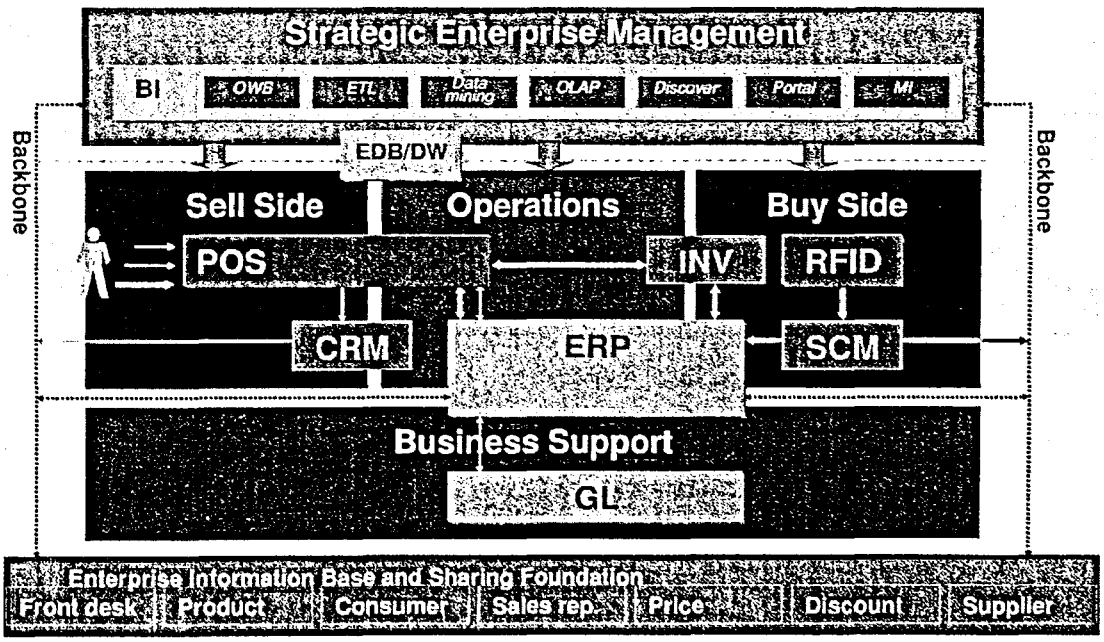


Example of DCRP in handling a use case

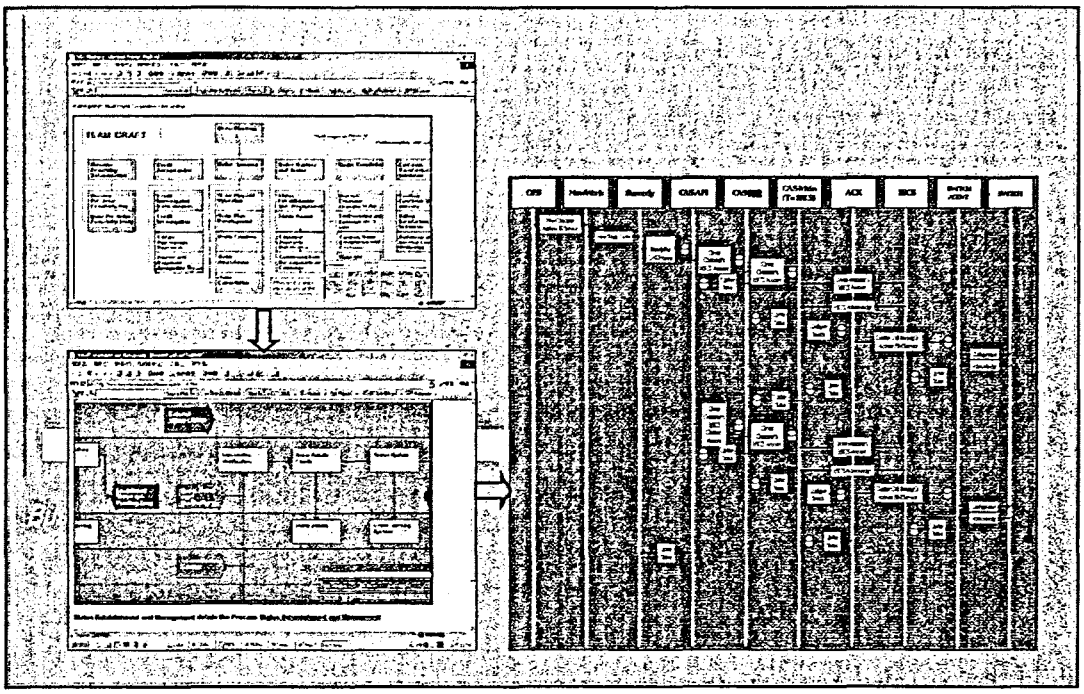


Benchmarking between Hash n:n, Hub & Spoke and SOA

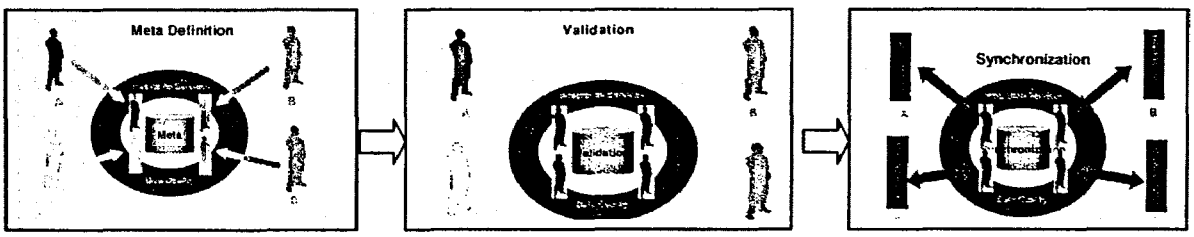




Business architecture decomposition



Metadata Transformation through Definition, Validation and Synchronization



S7

Impact analysis of the platform selection

Prepared by: Raymond Wu
Chief Architect
Vodafone Japan Global Planning Department

vodafone

for Distribution in Vodafone GITA

Preface

This analysis addresses the question and recommendation if we use VENDOR B hardware and operating system to support Billing system in CC&B , how much impact it may cause in both J-Phone implementation and target deployment if we transfer the technical resource to the other site or country
(please refer appendix 1 diagram to see the simple topology)

The investigation of adopting VENDOR B platform for the Billing system indicates that this is compatible with the generic Unix solution of the project as those software
(Portal Infranet, Siebel, Oracle ...), middleware (WebMethod, Tibco ...) and networking are non-platform dependent and fully transparent.

This report primarily tries to address the above feasibility, continue to pursue the best practice and minimize the impact it may cause if we choose a Hardware and Operating System of Vodafone non-standard and transfer the technology to the other site.
(Vodafone Global Architecture standard for Operating System is VENDOR A/Ux & Sun/Solaris Unix)

Recommendation

Feasibility

From Technology and Integrity perspective, both VENDOR A and VENDOR B platforms meet minimum requirements to support Portal Infranet's Billing system, and the
co-existence of VENDOR B in VENDOR A environment is feasible without measurable impact of overall operation. *(please refer appendix 2 diagram)*

Better Practice

However, the evaluation shows that VENDOR A provide better adaptability and Integrity to work with Portal Infranet and webMethods EAI, a remarkable benchmark in running Infranet and will be more suitable in J-Phone Billing system environment.
(please refer appendix 3 table, this table shows the overall evaluation)

Strategy

Strategy evaluation also shows that, VENDOR A/Sun platforms have been commonly used in Vodafone Global and has been the GITA recommended solution, the adoption of VENDOR A/Sun not only follow the GITA strategy, it will minimize the effort and risk of technology resource transfer between Vodafone OpCos.

Cost

From Cost perspective, a rough estimation of extra maintenance and training fee of 30% is required to put the VENDOR B operation into practice. These factors need be considered even it does not cover the extra interfacing effort and risk it may cause. In the other word, if the cost is not at least 30% lower from VENDOR B side, we may not be able to say that VENDOR B is a cost-saving option.

Case it applied

Finally, this evaluation is purely based on the Billing system platform selection. Other systems particularly the Enterprise and J-Phone Apac Data Center setup, need another assessments which VENDOR B solutions may be better applied in certain fields particularly those regional corporative or enterprise level. Therefore each consideration in non-standard platform adoption should be an individual case (form-follow-function), all such consideration need be flexible but need to refer to GITA Global strategy to minimize the impact of Global Architect.

Principles

The principles listed below are key guidelines for the design or purchase of applications and application components supporting distributed, client/server computing across the state.

Principle 1

- Design applications to be highly granular and loosely coupled
- The designer should allow for the possibility of re-partitioning an application in the future.
- Being highly granular and loosely coupled provide flexibility in physical implementation (i.e., in the deployment of application components on different platforms).
- Highly granular, reusable application components are key to increased productivity and rapid application deployment. The Componentware Architecture supports a highly granular design.

Principle 2

- Plan for extensibility and scalability
- Applications must evolve to support new business requirements and make use of new technologies.
- Extensibility provides functional scalability.

Principle 3

- Design application to reuse components
- Applications should be built by assembling and integrating existing components, rather than by creating custom code. Shrinking cycle times do not allow for artisan programming.
- Managing component reuse is supported by the Componentware Architecture.

1. **Technology and Integrity perspective**

   Both VENDOR A and VENDOR B solutions have capability to perform Billing System functionalities together with Portal Infranet, and co-existence of VENDOR B in VENDOR A environment is feasible by using Fibre Channel Hub without an impact of overall operation *(please refer appendix 2 diagram)*

   This evaluation shows that VENDOR A provide better Billing functionalities and Integrity to work with Portal Infranet and webMethods EAI, it has less Complexity and will be more suitable in J-Phone Billing system environment

   (please refer appendix 3 table, this table shows the overall evaluation)

2. **Strategy perspective**

   VENDOR A/Sun platform have been commonly used in Vodafone Global and has been the GITA recommended solution, the adoption of VENDOR A/Sun follow the GITA strategy and minimize the risk of technology resource transfer inside Vodafone OpCos

   (please refer appendix 3 table again)

3. **The evaluation is an individual case study, not for a generic purpose**

   This evaluation is purely based on the Billing system platform selection.

   Other systems particularly the Enterprise and J-Phone Apac Data Center setup, need another assessments which VENDOR B solutions may be applied in certain operational fields, this assure the best Vodafone practice and J-Phone operation.

   Therefore each consideration in non-standard platform adoption should be an individual case (form-follow-function), also all the consideration need be referred to GITA Global strategy to minimize the impact of Vodafone Global Architect

Summary

Selection Consideration

The evaluation and selection of Billing System Platform need be at least based on the following factors:

- Technology, Integrity and Strategy
- Budget and other special corporative level concerns

This document is focused on Technology, Integrity and Strategy factors which doesn't cover the budget or other special concerns, however the extra cost in VENDOR B system administration, training and maintenance will be a great amount. Therefore, the factor of budget or any special concern from management may need be further referred before coming to a decision. This evaluation work is purely based on the Portal Infranet Billing system platform selection inside CC&B which will be adopted in J-Phone, the recommendation should not affect or provide any hint in other system's platform selection.

List of the Summary

1. VENDOR A Opencall SS7 technology: information is automatically fed into the integrated Portal Infranet rating and billing application and has good Integration of Voice and Mobile Internet billing into a single Prepaid Balance solution which handle interfacing Infranet complex models efficiently.

   This function is believed to be compatible or more elegant than VENDOR B WebSphere Everyplace Suite or other VENDOR B eServer pSeries or MQ Series integrated eBusiness solution platform

2. VENDOR A OMI (Open Management Interface) integrated webMethods EAI provide a better

   function-enriched management interface and EAI event tracking and control than other platforms

3. On April 18 of 2000, VENDOR A 9000 Enterprise Servers became the first large-scale benchmark to run Infranet 6.0.1 on Oracle 8i release 2, the benchmark information is;
   - Massive Account Scalability (25 million user accounts)
   - High Speed Authentication (at least 4,500 authentication sessions per second)

- ✧ Authorization Capabilities (at least 255 million extended authorization sessions per day)
- ✧ High Speed Response (as low as 20 milliseconds per request)

Till now this benchmark has been the milestone of Infranet running on the large-scale platform,

The assumption is, VENDOR A platform, under the best practice configuration, provide the most effective performance on Portal Infranet better than any other platform or best-tuning solution

4. Both VENDOR A and VENDOR B has good solution for system monitoring and High Availability solution VENDOR B's HACMP high availability solution provide more advanced fail-over solution which need to handle the media failure or system corrupt, however, Billing system may not need the fail over solution as Siebel system was planned, so the fail-over redundancy should not be a concern at this stage
5. If for any reason, J-Phone adopt VENDOR B as it's Billing system instead of VENDOR A, the risks are
   a. J-Phone Billing system will risk the higher complexities in implementation, and potential performance loss
   b. J-Phone CC&B will encounter more difficulty in supporting or supported internal of OpCos
   c. J-Phone may need a stronger reason to support running a non-GITA strategy
6. VENDOR B platform co-exist with VENDOR A or Sun environment is feasible and should not be an issue in the transparency of distributed architecture. The adoption of VENDOR B platform or VENDOR B solution in other system particularly J-Phone APAC Data Center setup is possible, the feasibility need be addressed in Vodafone Global Forum in compliance of Global Operation

Technology Consideration

1. Infranet Portability

Portal Software, Inc. and Hewlett-Packard Company have partnered to extend Portal's convergent wireless customer management and billing capabilities to the prepaid voice market. VENDOR A Opencall delivers SS7 technology for single prepaid balance. Competing solutions generally require two distinct systems to handle prepaid voice and mobile Internet billing—resulting in two separate and inconvenient user account balances and higher customer management costs—or expensive integration with legacy IN systems. The integrated Infranet and VENDOR A Opencall solution consolidates voice and mobile Internet activity into a single prepaid balance. The Infranet platform offers open APIs and sophisticated remittance functionality to support even the most complex business models in the mobile Internet value chain. Infranet includes a prepaid manager to handle interaction with VENDOR A Opencall. By combining elements Portal's Infranet and VENDOR A Opencall Intelligent Network (IN) Platform (a real-time call-processing platform), mobile operators can deploy a state-of-the-art, end-to-end prepaid billing infrastructure solution comprising billing software, service control point software, hardware, and systems integration services.

VENDOR B's WebSphere Everyplace Suite

VENDOR B Websphere integrating Infranet for wireless service providers that have begun deployment of next-generation mobile Internet services. The alliance leverages VENDOR B's extensive product portfolio of carrier-grade servers and middleware in addition to VENDOR B's renowned industry-specific professional services. Portal Software's experience with supporting next-generation services such as digital content and rich voice forms the foundation of the joint 3G-ready solution.

The integrated solution will enable personalized, next-generation services capable of generating immediate revenue while accelerating deployment of new mobile wireless services. The WebSphere Everyplace Suite will enable service providers to create mobile Internet services with advanced software elements such as transcoding, message queuing, synchronization, and security in addition to core technology enablers such as a wireless gateway capable of supporting multiple networks and protocols. The above features provided by VENDOR A and VENDOR B regarding Infranet Portability, it's believed VENDOR A is more mature and one further step advanced in the two way synchronization and real time billing

2. EAI interfacing and Networking

webMethods, Inc. and Hewlett-Packard Company have announced the public availability of the Open Management Interface (OMI) Specification Version 1.0 which defines a standards-based (XML/SOAP/HTTP) management interface. With OMI, users will be able to use systems management tools to effectively manage and monitor business processes, web services and the underlying integration infrastructure that drives their businesses. To Manage Universal Application Network Business Processes along with Underlying Hardware and

Software.

VENDOR B

VENDOR B and WebMethods has not formed any partnership like VENDOR A did, and both companies became competitors in the IT industry, therefore, the implementation of WebMethods on VENDOR B platform is feasible as WebMethods running the industry standard like. HTTP and XML, but as VENDOR B and WebMethods has not developed any commercial product like VENDOR A's OMI, the interfacing is stay in the basic level

VENDOR A and WebMethods developed OMI for better message control, however VENDOR B and WebMethods Still stay in a protocol basic level

3. Platform Benchmark

Benchmark of Infranet 6.0.1 on VENDOR A 9000 Enterprise Servers and Oracle 8i (on April 18, 2000)

- Massive Scalability (25 million user accounts)
- High Speed Authentication (more than 4,500 authentication sessions per second)
- Authorization Capabilities (more than 255 million extended authorization sessions per day)
- High Speed Response (20 milliseconds per request)

VENDOR A 9000 Enterprise Server was the first large-scale benchmark to run Infranet 6.0.1 on the Oracle 8i release 2, and is the first Infranet 6.0.1 benchmark to run on the 32-CPU VENDOR A 9000 V2500 Enterprise Server. This unprecedented test demonstrates that Infranet can scale to 25 million user accounts. In addition, the ITM-C benchmark shows real-time online transaction processing performance of more than 4,500 authentication sessions per second, or 2,900 authentication and authorization sessions per second, demonstrating Infranet's ability to handle high priority processes rapidly and effectively. In addition to scalability tests based on user accounts totaling 25 million, ITM-C simulates processing of customer logins, including a real-time status check of customer accounts. Infranet performed login tests at a rate of nearly 16 million per hour. Infranet also performed at a rate of more than 255 million extended authorization sessions per day. Each of these operations included a simultaneous credit limit check on each account, which is only possible with Infranet's unique comprehensive real-time processing capabilities.

Other ITM-C tests show excellent customer service request response and bandwidth. Infranet performed more than 60,000 service representative requests per minute with average response times of only 20 milliseconds per request, confirming Infranet's ability to handle a huge number of service requests.

VENDOR A platform was the milestone of Infranet Performance Benchmark in AAA and TX response

4. System Monitoring & High Availability Tools

VENDOR A Event Monitoring, High availability and fail over monitoring

The VENDOR A OpenView Portal SPI integrates Portal Infranet into the rest of the IT environment managed by the VENDOR A OpenView family of products. It is preconfigured to easily install and link with the OpenView console to extend the event-management and process-monitoring capabilities of Portal Infranet. It perform the following tasks;

- ✓ Monitors the Portal intranet application server (log file)
- ✓ Checks for continuous service availability using the testnap probe
- ✓ Monitors CM, CMMP, CM, JS, and Radius critical processes

VENDOR A MC/ServiceGuard is an high availability solution that provides automatic fail over capabilities for clustered VENDOR A 9000 Enterprise Servers. It is a specialized facility for protecting mission-critical applications from a wide variety of hardware and software failures. With MC/ServiceGuard, up to 16 nodes are organized into an enterprise cluster that delivers highly available application services. VENDOR A MC/ServiceGuard monitors the health of each node, and quickly and automatically responds to failures in a way that minimizes application downtime. When Infranet is operating under the MC/ServiceGuard environment, most conditions that threatens its availability will trigger the automatic transfer of applications to other enterprise servers. This high level of availability is critical to service providers to keep their mission-critical applications up and running. Customers can define conditions under which Infranet applications will be transferred to alternate fail over VENDOR A9000 enterprise servers. These conditions not only include system processor, memory and disk failures, but also failures in network media, adapters, system and application processes.

VENDOR B eServer pSeries and HACMP solution (high availability)

Portal announced in October 2001 that Infranet is available on VENDOR B eServer pSeries enterprise UNIX servers. The pSeries supports mission critical e-business applications and provides Portal and VENDOR B customers with key performance, scalability, availability and support advantages. VENDOR B is well-known for it's high availability product (HACMP) which supports 24x7 availability of Check Point FireWall-1 Switch over of Check Point FireWall-1 gateway on node failure is transparent to network users (automatic transfer of IP address to healthy system). There is no routing or network reconfiguration required, when system crash, all existing connections (telnet, ftp, http) going through Check Point's FireWall-1 Gateway survive switch over. AIX HACMP takes very little time to switch over. And is flexible to configure - can handle availability of applications, disk and network. It has proven industry solution with many industry installations.

Regarding system monitoring, the VENDOR A Openview Infranet monitoring may be closer to J-Phone Billing system's need, VENDOR B provide some monitoring tools (like Tivoli) or VENDOR B eServer pSeries are compatible but this leave the dependency on platform version selected. Regarding the high availability, VENDOR B's HACMP is well-known for it's seamless solution and more advanced than VENDOR A's MC/ServiceGuard in the fail over automation, however this feature may not required in J-Phone promptly, or if it's required in the future, the third party software providers can enhance this part.

Strategy Consideration

GITA Compliance strategy

The purpose of Vodafone Global IT Architecture Framework is to support global business requirements while enabling global synergies through the definition of a common reference framework for the development of IT Architecture across Vodafone group. The strategy in Global Architecture guidelines indicate that VENDOR A and Sun platform are the recommended hardware, although GITA doesn't limit any brand of platform which Vodafone should not use, however, if J-Phone Billing System adopt VENDOR B platform, it will weaken the compliance of J-Phone with GITA Global strategy

J-Phone IT Architecture Integrity strategy inside CC&B

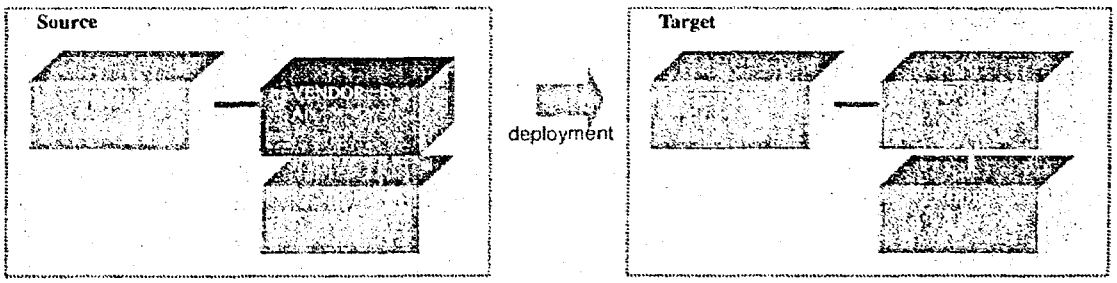
Billing inside CC&B is highly integrated with other system (CRM, Rating, Account mgt, Product mgt...),

(please refer appendix 5 diagram).

when other VENDOR A or Sun system, which has interface with Billing system, use certain package or feature which cannot be performed in VENDOR B environment (i.e.; Weblogic is used as web engine in Sun or VENDOR A, which may have difficulty in interfacing the Websphere in VENDOR B machine, will cause a problem, or if we skip Websphere from VENDOR B machine to prevent this problem, then the VENDOR B solution may loss the best practice solutions we need), therefore the adoption of VENDOR B machine is suggested to be considered for those less interface area from integrity perspective to maintain a less-heterogeneous interface between the pac
`11 kage software which running efficiently in either VENDOR A or VENDOR B environment.

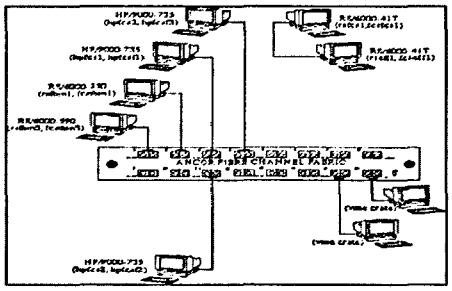
Appendix 1

Diagram show the technical resource transfer (OpCos internal deployment) between different site or country with different hardware platform between the sites



Appendix 2

Heterogeneous Fibre Channel Architecture example. Overview of the different types of interfaces and Fibre Channel equipment available at CERN



Appendix 3

Evaluation - Overall Rating

Consideration	Evaluation	Remark
VENDOR A solution	Recommend (2.5 ~ 3)	better technology capability and maturity, better integrity, less complexity and high Vodafone GITA compliance
VENDOR B h/w + planned package	Feasible (2)	marginal technology maturity and strategy but feasible
VENDOR B solution	Not recommend (1.5 ~ 2)	acceptable or good technology capability, but high complexity inside CC&B and weak in integrity and GITA compliance strategy

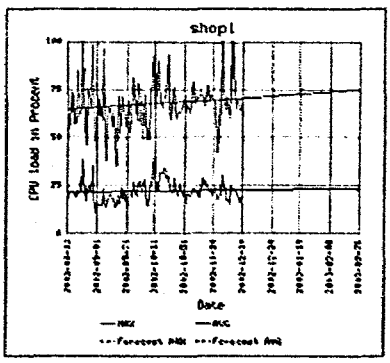
Evaluation - Item Rating *(3: good, 2: acceptable, 1: marginal or not feasible)*

Consideration	Evaluation Item	VENDOR A	VENDOR B h/w + planned	VENDOR B
Technology	EAI interfacing	3	1 ~ 2	2
	Intranet portability	2~3	1 ~ 3	2~3
	performance benchmark	3	1 ~ 3	2 ~ 3
	system Monitoring	2	1 ~ 2	2
	high availability	2	1 ~ 3	2 ~ 3
Integrity, Complexity	depending on the case	3	2	1

Example of the Best Practice of VENDOR B vs VENDOR A solution in the Billing system

Appendix 4

The operation of VENDOR A OpenView at Vodafone D2 GmbH



The most important applications for Vodafone D2 are running in a clustered SUN Solaris (2.6) environment. Each cluster is responsible for a different set of applications. Our mointoring group is monitoring these servers using VENDOR A OpenView agents. This provides us with detailed and accurate performance data about our servers.

Currently the rollout of the central storage process for the measured data is not complete. So we can just see a subset of all servers in the central monitoring database (called MDCB).

The VENDOR A OpenView agents are storing this data also locally. Therefore after the rollout is complete we will see all servers in the MDCB with all their prior measured data.

The MDCB provides our group with data about CPU, memory and disk usage. As the data is much to detailed to be used directly we have to condense it.

Currently we are planning to develop a web based application to provide this information to other groups. This application will be capable to forecast the load of a certain machine for a given time frame. This forecast will be based on the historical data in the MDCB.

Contingency

The Contingency requirements in Billing system or some area in Customer Care can be higher than regular system. Automatics in Monitoring, Detection and Auto Switch-over may not required at the beginning, but Contingency plan such as; Primary Server (DB) and Secondary Server (DB) with Warm-standby and robust Switch-over mechanisms are definitely required in Billing and some area of the Customer Care systems.The scenarios of switch-over is when primary server is down or encounter an outage during the batch process or need maintenance that cannot handle the user process and need the users be re-routed to the secondary server. When primary server become inactive, the secondary server acted as a new access point for possibly query-only service, all the non-query transaction need be stored in a staging area and wait until the primary server back to service, these stored transaction records then need be applied or sent to primary server after switch back.

Batch Process

Siebel/Portal Batch process mainly perform upload and download, combined with the daily batch process from OS level, the primary server need to switch the user access to secondary server. Batch process need an assessment for using Oracle partition or special Siebel/Portal configuration by using a code to flag the record for processing to eliminate the workload, so batch process run against only those small portion instead of a full-table-scan. Archiving by using Ju-box also need be considered for performance improvement. The secondary database need be fully utilized, so during the server outage or heavy

Batch process, the secondary server need to take over the access control and handle the user request until system switch the access control back to primary.

System Architecture & Integration

Siebel and Portal Package

Need to review those requirements that the Siebel or Portal packages cannot handle and need be customized. I would like to suggest the minimizing of such customization work to less than 10%. (this value stands 30% in the current Accenture's investigation)

Even Siebel said 70% (or higher) of standard Siebel solution is reasonable, I think at least 85% of standard Siebel solution is required. Please list and brief the out-of-Siebel-scope requirements so we can review them and decide the solutions. Example is the joining account case that Accenture raised in the meeting, the possible solution is; to apply and associate service sets from both family and organization to the account which need be beneficial from both sources, instead of changing the account hierarchical structure. The other possibility is applying profile to those account which need be impacted. Standard Siebel and Portal solutions doesn't guarantee the easier process or performance gaining in real-time & batch process, however I think we need to use these standard solution as much as we can, and tune the performance issues in each layer, this is one of the reason I am considering Parallel server to work on these issues. Join layer between Siebel and Portal (UI, Services, Data layers...) need a strategy analysis for how to handle the common functionalities, examples are; Authentication, Authorization, Product/Account validation, sync .... Either Siebel or Portal may handle these processes differently, which side need be activated as primary is a concern.

Data Architecture

Master DB

The pattern for Master DB decision is;
Master DB = Siebel DB + Portal DB + (a reasonable customized schema sets)
Regarding what portion can be the customized need a review and agreement.

According to Vodafone Global Architecture principles, both Siebel and Portal
can co-exit in one database or separately exist in different database, but in either case data duplication is not allowed. The other requirement is Siebel standard Enterprise
solution and database need be deployed as a priority;

This enhancement need to work together with the Contingency plan which include;

1. Primary and Secondary warm-standby database architecture
2. Oracle Parallel servers architecture is recommended for DB transparency
3. Possibly Staging and Oracle partition solutions need a feasibility study
4. Need to set the cut-off time for switch over, all the transactions happen after cut-off need be processed in the second day
5. Need a robust switch mechanism (i.e.; use initORA files...) to switch user access between primary and secondary. So prim can switch the access control to secondary at cut-off, and switch back after prim complete the daily (batch) process
6. Secondary DB may provide query-only services, may need a staging area (DB schema set) to take the transaction when primary become invalid, once prim complete the Batch job, the staging area sending the pending transaction to primary to roll the primary DB forward, system switch the access from secondary back to primary, and synchronize the prim to secondary DB by physical file copy.

Infrastructure

Topology

Parallel servers is recommended to handle Siebel/Portal for both Contingency and Performance purpose (refer Ref.2), this need to use Clustering and Distributed technologies as the foundation. Oracle Parallel servers provide Oracle Instance on the server allow users to connect with a particular Oracle database as local database user, Oracle parallel server also provide clustering and distributed features that the Database servers are fully transparent with users.

S8

High performance system design using the combination of technical solutions

IBM

Presentation to IBM ASEAN SWITA

Raymond Wu
IBM Software group
June 2006

Abstract

This paper mainly covered an investigation of system performance issues by combination solutions of event driven time-volume control, meta cursor computing, parallelism, database partition and archiving, the investigation was involved in large scale distributed within industry independent environment.

In the following paragraph, the sample codes and benchmarking result were provided to reflect cache memory utilization, hit ratio of logical read/write over physical read/write, advanced SQL query, meta approach, partition and data classification. As data aggregation is important in creating metadata in supporting business integrity and system performance, the scenario in this investigation also taking data aggregation into consideration.

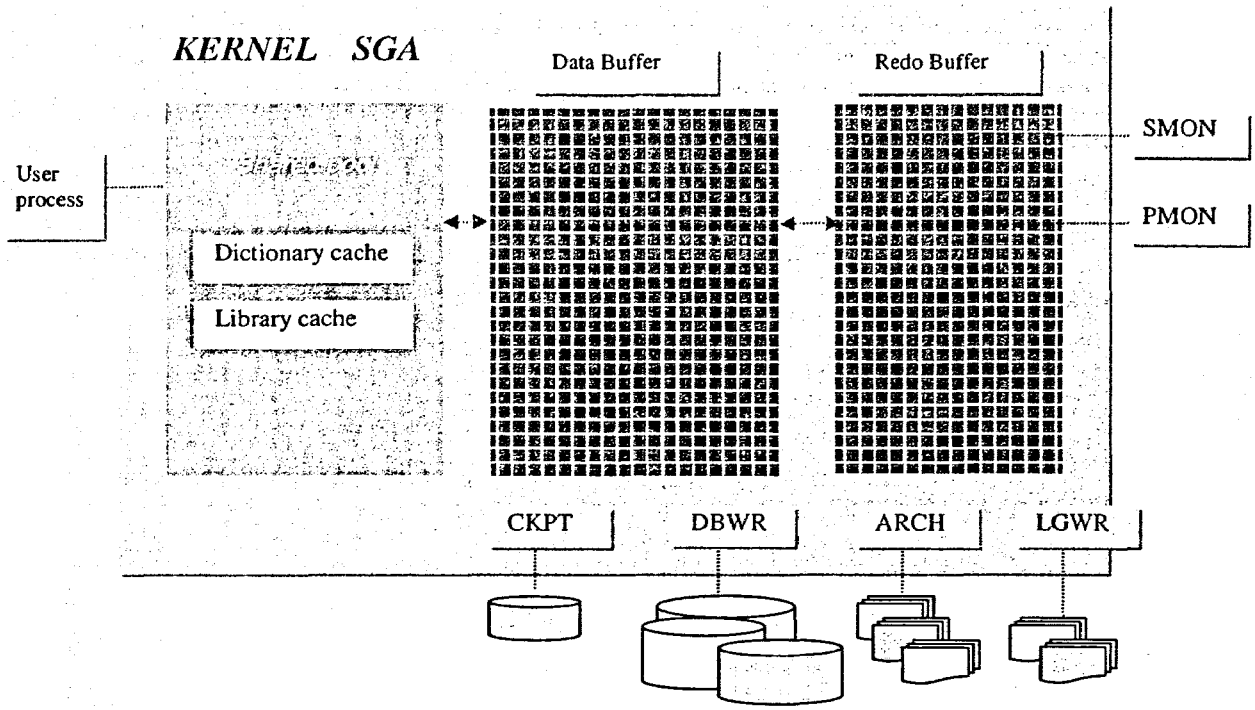
Event driven time-volume engine design

In a real-time and batch process co-exist environment, event driven approach was tried to work on the optimization between volume and time.

The situation is response time is not critical for many real-time process which consuming huge system resources in query and update of database for each record, the approach set both time and volume limit for incoming event, so when the incoming event reach the limit of either volume (i.e. 100 records) or time (i.e. 5 minutes) the trigger will be fired and the accumulated records will be submitted for further processing. Event driven technology also allow particular event be pre-configured, so once the condition met, the trigger will be trigged in the same manner of time and volume.

Event driven technology handle real-time process by pre-defined conditions in the manner of batch, we can pre-define the parameters by taking maximum allowable time and volume, and pre-define conditions to submit only once the parameters or conditions met.

During the business peak hours, it become easier to reach the volume high limit and takes shorter period to reach the limit, in the reverse case when the incoming event become slow in the midnight, the pending records may wait until the time limit is triggered to submit the batch.

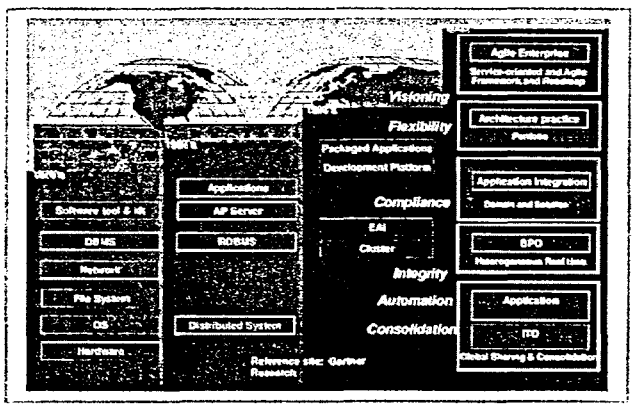


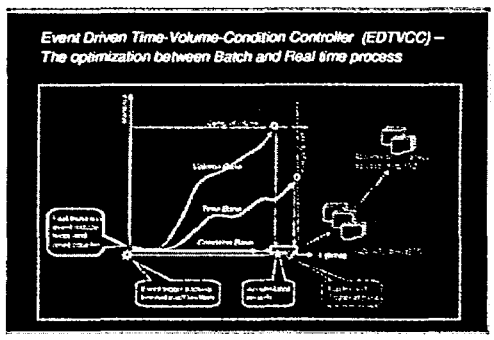
Following enhancement is based on the technical implementation of following solution assets -

- Event driven by time-volume control
- Meta data
- Cursor computing
- Parallelism
- Data partition and archiving

Following diagram briefly introduce the trend in these 30 years.

In this IT new age, Agile, Enterprise, Integration are all the priority and focus that need robotic computing and combined solution to solve the performance issues.





Metadata

Metadata is the aggregation or definition of data, it can be in the form of information driver, data aggregation, a script or description of the raw data.

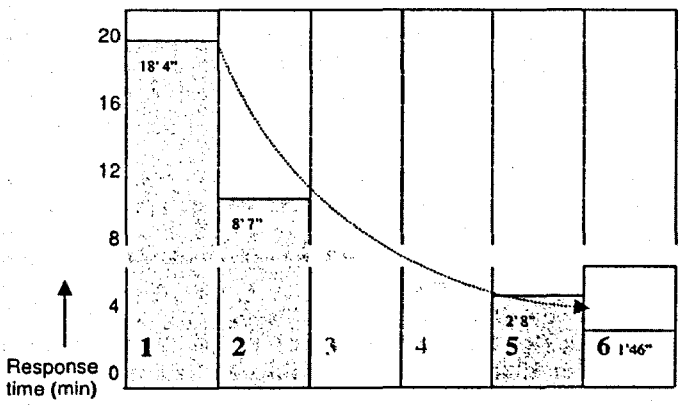
Data Aggregation

Metadata computing can be used as an efficient way to synchronize between databases, or symbolic "highly condensed" information for an easy retrieval of identified records. In an master data environment or centric database, all the distributed database can use self-generated script for remote master database to run the scripts on global database as a single source of truth, so each peripheral local database doesn't need to send high volume of data to manipulate its global database, the only update is a small file of either runtime script or identified records with its record key, this save huge network and database load. Metadata in the form of aggregated information, can be a good asset of reuse object, so several interested fields of information can be aggregated toward metadata for further computing, so when statistics again metadata for particular dimension save computing resource as it doesn't need to retrieve huge volume of raw data each time, while retrieval of these raw data may cause locking issue which is the other advantage of metadata computing.

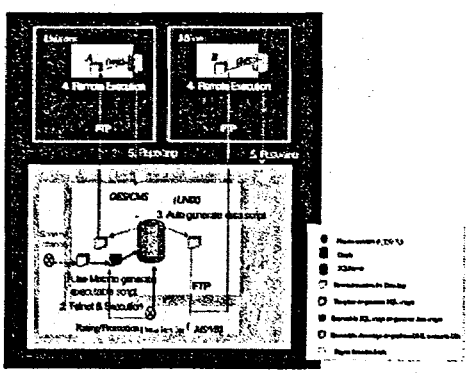
Query Student in Class_code=10	Query Student in Class_code=10	Query Student in Class_code=10
Without Index => *runs in 55 seconds*	**With Index on student** => *runs in 70 seconds*	**With concatenated index on student and class** => *runs in 0.1 second*
Execution Plan	Execution Plan	Execution Plan
0 SELECT STATEMENT Optimizer=CHOOSE 1 0 TABLE ACCESS (FULL) OF 'EMP'	0 SELECT STATEMENT Optimizer=CHOOSE 1 0 TABLE ACCESS (BY ROWID) OF 'EMP' 2 1 INDEX (RANGE SCAN) OF 'EMP_DEPTNO' (NON-UNIQUE)	0 SELECT STATEMENT Optimizer=CHOOSE 1 0 INDEX (RANGE SCAN) OF 'EMP_DEPTNO_ENAME' (NON-UNIQUE)
Statistics	Statistics	Statistics
0 recursive calls 2 db block gets 219 consistent gets 7 physical reads 15654 bytes sent via SQL*Net to client 1155 bytes received via SQL*Net from client 83 SQL*Net roundtrips to/from client 0 sorts (memory) 0 sorts (disk) 1203 rows processed	0 recursive calls 0 db block gets 2408 consistent gets 0 physical reads 15654 bytes sent via SQL*Net to client 1155 bytes received via SQL*Net from client 83 SQL*Net roundtrips to/from client 0 sorts (memory) 0 sorts (disk) 1203 rows processed	0 recursive calls 0 db block gets 94 consistent gets 0 physical reads 15654 bytes sent via SQL*Net to client 1155 bytes received via SQL*Net from client 83 SQL*Net roundtrips to/from client 0 sorts (memory) 0 sorts (disk) 1203 rows processed

Cursor computing

Cursors are database objects used to traverse the results of an SQL query. Cursor computing reserve a computing area in memory for record base computation, so in a recursive nested computing scenario, it gain the fasted response as table join is taken placed in cache memory instead of database, the higher complexity of table join and logic check, the higher resource saving as once the table records are read into the cursor memory, all the computation effort within memory can be neglected.



method (* refer next table for method number)



Record size = 2,000,000 (buffer =1,000,000 bytes)

1. Traditional SQL	18' 3.59"
2. Hint enhancement:	8' 7.18"
3. /*+FULL*/	5' 34.79"
4. /*+INDEX*	3' 40.94"
5. Cursor Programming Type A (set filter in query)	2' 8.04"
/*+ORDERED*/	2' 9.33"
/*+ALL_ROWS*/	2' 24.92"
/*+FULL(T32)*/	2' 16.04"
6. Cursor Programming Type B (set filter in cursor)	1' 46.32"

Code used in benchmarking

*** Traditional SQL (Group 1)

```
SELECT
A1.CHG_ID,A1.REQ_TYPE,A1.CONTRACTNO,SUBSTR(A1.CONTRACTNO,
1,2),
A1.TASK_ID,A1.TASK_NAME,A1.ASSIGNED_TO,A1.TASK_STATUS,A1.RO
WID
FROM ARADMIN.A_TASK A1    WHERE A1.TASK_STATUS='1'
AND              (SUBSTR(A1.CONTRACTNO,1,2)='CT'              OR
SUBSTR(A1.CONTRACTNO,1,2)='GS')
AND (A1.ASSIGNED_TO='NOC' OR A1.ASSIGNED_TO='NOC1')
AND (A1.REQ_TYPE='REQ_025016'
    OR A1.REQ_TYPE='REQ_025006'
    OR A1.REQ_TYPE='REQ_025015'
    OR A1.REQ_TYPE='REQ_025014'
    OR A1.REQ_TYPE='REQ_025005'
    OR A1.REQ_TYPE='REQ_025013'
    OR A1.REQ_TYPE='REQ_025012'
    OR A1.REQ_TYPE='REQ_025008'
    OR A1.REQ_TYPE='REQ_001019'
    OR A1.REQ_TYPE='REQ_001014'
    OR A1.REQ_TYPE='REQ_001015'
    OR A1.REQ_TYPE='REQ_001018'
    OR A1.REQ_TYPE='REQ_001012'
    OR A1.REQ_TYPE='REQ_001013'
    OR A1.REQ_TYPE='REQ_001022'
    OR A1.REQ_TYPE='REQ_001021'
    OR A1.REQ_TYPE='REQ_001101'
```

*** Hint enhancement (Group 2)

```
SELECT /*+ALL_ROWS*/
A1.CHG_ID,A1.REQ_TYPE,A1.CONTRACTNO,SUBSTR(A1.CONTRACTNO,1,2)
,
A1.TASK_ID,A1.TASK_NAME,A1.ASSIGNED_TO,A1.TASK_STATUS,A1.ROWID
FROM ARADMIN.A_TASK A1   WHERE A1.TASK_STATUS='1'
AND              (SUBSTR(A1.CONTRACTNO,1,2)='CT'              OR
SUBSTR(A1.CONTRACTNO,1,2)='GS')
AND (A1.ASSIGNED_TO='NOC' OR A1.ASSIGNED_TO='NOC1')
AND (A1.REQ_TYPE='REQ_025016'
    OR A1.REQ_TYPE='REQ_025006'
    OR A1.REQ_TYPE='REQ_025015'
    OR A1.REQ_TYPE='REQ_025014'
    OR A1.REQ_TYPE='REQ_025005'
    OR A1.REQ_TYPE='REQ_025013'
    OR A1.REQ_TYPE='REQ_025012'
    OR A1.REQ_TYPE='REQ_025008'
    OR A1.REQ_TYPE='REQ_001019'
    OR A1.REQ_TYPE='REQ_001014'
    OR A1.REQ_TYPE='REQ_001015'
    OR A1.REQ_TYPE='REQ_001018'
    OR A1.REQ_TYPE='REQ_001012'
    OR A1.REQ_TYPE='REQ_001013'
    OR A1.REQ_TYPE='REQ_001022'
    OR A1.REQ_TYPE='REQ_001021'
    OR A1.REQ_TYPE='REQ_001101'
    OR A1.REQ_TYPE='REQ_001102'
```

Parallelism

Parallelism split a huge task into several small pieces of tasks so that several processes share the load from these small tasks and do part of the that huge work. In the system design, its usual that multiple CPUs handle their portion simultaneously, significant performance gains can be expected.

Data partition and archiving

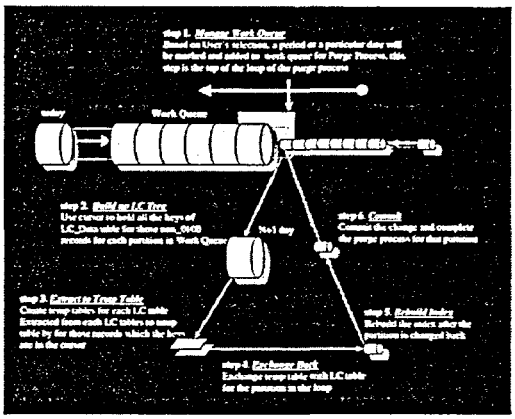
Partitioning enhance the performance of variety of applications, it allows tables, indexes, and index-organized tables to be subdivided into smaller pieces, enabling these database objects to be managed and accessed at a finer level of granularity to address every business requirement. Each piece of database object is called a partition. Each partition has its own name, and may optionally have its own storage characteristics, such as having table compression enabled or being stored in different tablespaces. From the perspective of a database administrator, a partitioned object has multiple pieces which can be managed either collectively or individually.

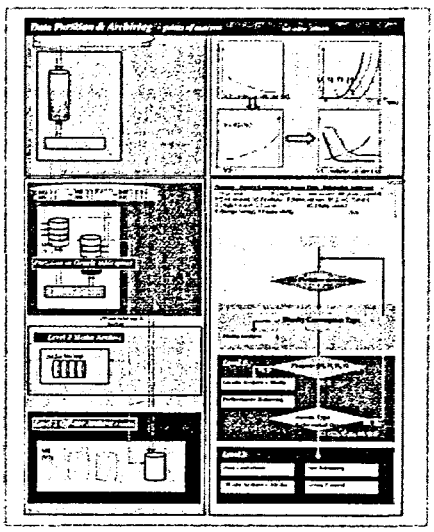
Partitioning improve query performance by several orders of magnitude. For example, suppose an application contains an customer master data containing a historical record of customer, and that this table has been partitioned periodically (i.e. by week). A query retrieving a single week would only access a single partition of the customer data. If the Orders tables had 1 year of historical data, this query would access one partition instead of 52 partitions.

The optimization strategy of how refine we partition the data depend on the nature of query and archiving, if we define the data quality by using the evaluation of business critical, life cycle status and query frequency, we can concentrate all "pending" status records in one partition for those outstanding services, we can partition the data on a monthly basis instead of weekly basis if the data volume is small, or the batch process is based on a monthly basis and much more critical than those data query in daily activities.

Database partition strategy

We can also partition the data based on the business status or history, and archive the data once the status become "inactive" or less critical in the query basis. In such case, archiving of inactive or lower quality to a less expensive media happen in less a second, the advantage from enterprise computing economy are; 1) keep the volume of higher quality of data as small as possible and be focused in small partition(s) help the system performance and response time, 2) simplify the application process as once the "active" or higher quality data are filtered out from the master data, the application can apply simple sql-based "mass change" instead of volume of "business logic" which save significant application design effort and run-time performance, 3) IT department can save the cost of storage as once the lower quality of data or "inactive" data are filtered out



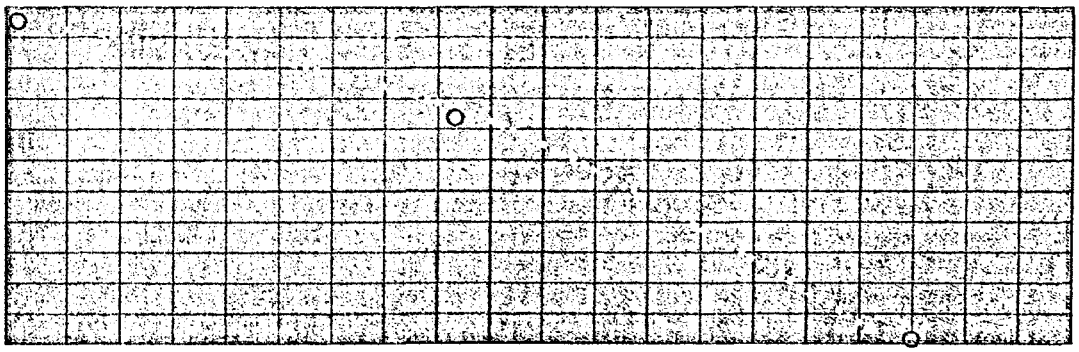


Benchmarking the partition scenarios

This test investigate the relationship between response time and Partition data retention. Partition retention means the percentage that queried records over the total record number, table used in testing: Testing table with 760,000 records

- Without partition (100% retention): Response time: 15'
- Partition (80% retention): Response time: 12 seconds
- Partition (50% retention): Response time: 10 seconds
- Partition (40% retention): Response time: 8 seconds
- Partition (20% retention): Response time: 6 seconds
- Partition (2% retention): Response time: 3.5 seconds

This fundamental testing show that when we concentrate the data into 3% (by using partition) instead of original 100% (without partition), the performance can be improved by a satisfactory factor



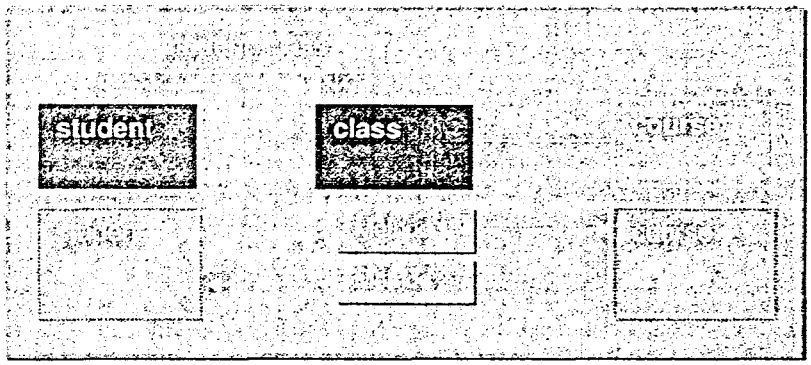
100 *(process partition retention %)* 50

Data archiving

Data archiving is the process that the non-active or less critical data to be archived to either on-site partition for reference or off-site media. The off-site media is for future reference only, it doesn't provide a convenient access. Both of on-site or off-site archiving are handled by data partition as in a production environment, particularly huge database scenario, it's impossible to submit a query and retrieve all the archived data into an archived area which cause high impact and take very long. The feasible way is using periodical partitioning and as the command happen within 1 second to "detach" the archived partition, this archiving process can be executed periodically in a very efficient way without causing any operational impact, once the database archives its non-active partition, database gain performance due to the smaller size. So both operational archiving and run-time issues can be resolved.

Meta Information Driver

Decide the meta information as driving force achieve better performance in a join query



Scenario 1

10,000 students (each student takes 5 courses)
1,000 course (each class has 25 students)
The direction will be decided based on the index on the student_class table:

- (student_Id, class_Id) forces student to be the driving table
- (class_Id, student_Id) forces course to be the driving table (this what we want)
- course : class= (1000 * 25) : (10000 * 5)=1 : 2

Query

Select all students that take "IT Fundamental"

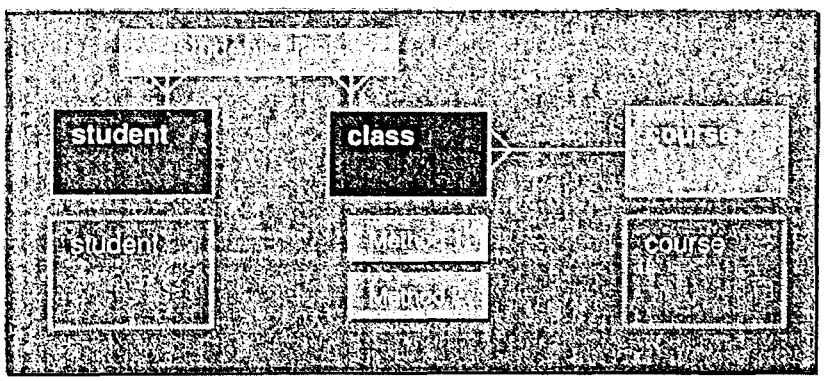
Benchmarking

Method 1. Use student as driving table (student -> student_class -> class -> course)

- Total accesses: 10,000 + 2 * 25 + 2 + 1 = 20,053

Method 2. Use class as driving table (course -> class -> student_class -> student)

- Total accesses: 1 + 2 + 2 * 25 + 50 = 103



Scenario 2
10,000 students (each student takes 5 courses)
1,000 course (each class has 25 students)
The direction will be decided based on the index on the student_class table
- (student_Id, class_Id) forces student to be the driving table
- (class_Id, student_Id) forces course to be the driving table (this what we want)
- course : class= (1000 * 25) : (10000 * 5)=1 : 2

Query
Select all courses that student "raymond wu" is taking

Benchmarking
Method 1. Use student as driving table (student -> Student_class -> class -> course)
- Total accesses: 1 + 5 + 5 + 5 = 16

Method 2. Use class as driving table (course -> class -> student_class -> student)
- Total accesses: 1000 + 2000 + 5 + 1 = 3006

The operation of HP OpenView at Vodafone D2 GmbH

The most important applications for Vodafone D2 are running in a clustered SUN Solaris (2.6) environment. Each cluster is responsible for a different set of applications. Our mointoring group is monitoring these servers using HP OpenView agents. This provides us with detailed and accurate performance data about our servers.

Currently the rollout of the central storage process for the measured data is not complete. So we can just see a subset of all servers in the central monitoring database (called MDCB).
The HP OpenView agents are storing this data also locally. Therefore after the rollout is complete we will see all servers in the MDCB with all their prior measured data.

The MDCB provides our group with data about CPU, memory and disk usage. As the data is much to detailed to be used directly we have to condense it.

Currently we are planning to develop a web based application to provide this information to other groups. This application will be capable to forecast the load of a certain machine for a given time frame. This forecast will be based on the historical data in the MDCB.

Conclusion

There are two major areas in software development life cycle which are related to system performance, they are technical design and operational run-time concerns. In this research paper, both of them were addressed. Technical implementation, which take the output from business modeling and works on the optimization of performance and integrity.

Within the technical design, there are abundant methods and technologies used in performance enhancement which may include, hardware, software, firmware, tuning and system design, while operational design impacts the system performance from corporative perspective, it mainly cover operational scheduling, strategy and topology.

This research paper raised the following questions to be discussed and further research;
1. What is the nature of system performance and impact factors
2. Is there any combination solutions that significantly improve performance rather than individual solution of performance improvement
3. What is the fundamental mechanism that combination solution improve the performance

S9

Telecom Architecture and eTOM

ORACLE

Agenda

1. eTOM Introduction
2. A brief mapping of AP industry to eTOM
3. The major tasks of Telecom Architecture
4. Architecture Principle
5. Open discussion

Facilitator:

Raymond Wu

Lead Consultant, Oracle Corporation

May 2003

Presentation for Telecom Industry in Taiwan

eTOM (enhanced Telecom Operation Map)

1. Architecture is the most efficient way to establish or expand the Business Domain particularly in the Telecom industry.

2. Architecture include; Business Architecture and System Architecture.

3. The basic definition of Architecture is; the interference and relationship between Objects or Components.
   The study, analysis and Design for these interference.

4. Most of the Architecture Methodology start from a robust Business Architecture tool for Business Modeling with certain initial involvement of System Architecture work to study the Infrastructure, the study provide the strategy of 'Buy and Build' of Infrastructure.

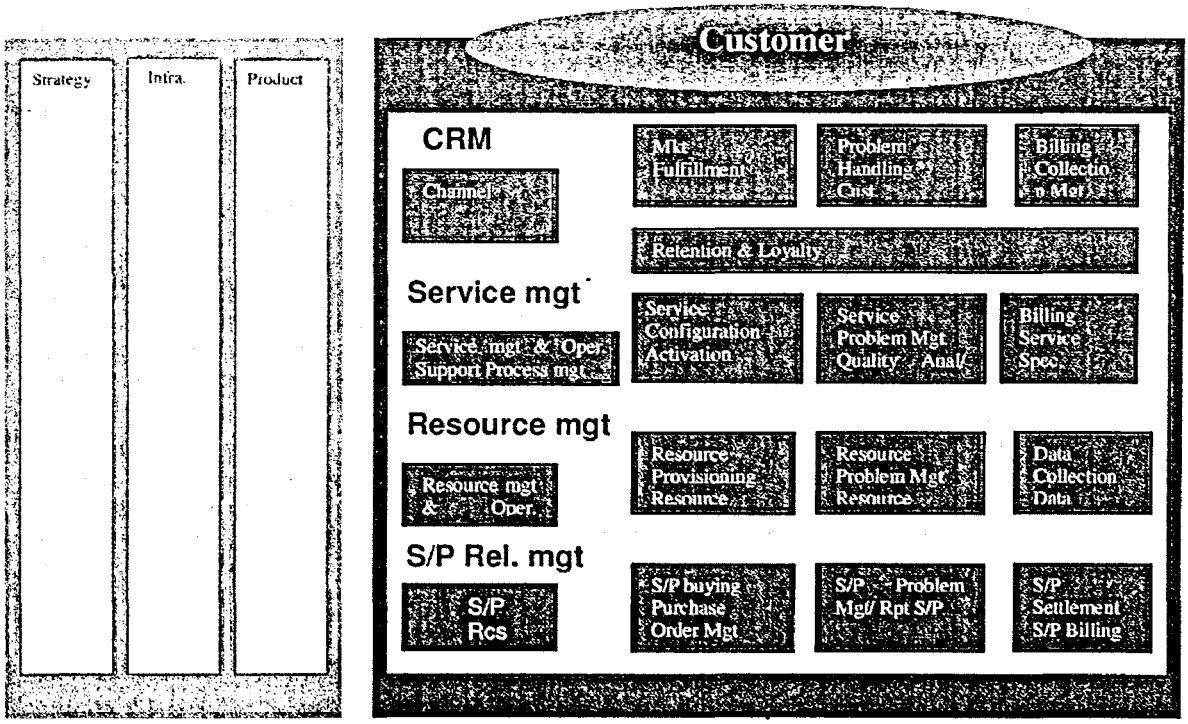
5. Once Business Architecture completed Business Modeling, System Architect will see the output for those Business Component and Flow.

6. System Architect need to review Business Architecture work to decide the Capacity Plan for Infrastructure strategy and basic System functionalities. Need to decide the primary Process functionality, Process Flow, Key Data model and Data Flow.

7. It is recommend to use a single Data Source (Master DB) at the enterprise level, Data flow, and component boundary need be minimized to prevent the system complexity.

8. In the World class IT standard, many of the individual components in the same layer are integrated as a common component (i.e; OSS, BSS, SDP ...) this ensure the industry standard and enhance the collaborative compliance between company and industry.

eTOM level 1



Reference site:

eTOM (enhanced Telecom Operation Map): http://www.casewise.com/extra/etomsample/di113.htm
TM (TeleManagement Forum): http://www.tmforum.org/browse.asp?catID=414

What Need be Standardized: (sample only)

- Business Model
- Tools
- Coding Standard
- Coding Language
- Terminology
- Data Model, Data Flow and method for Data Manipulation
- Process Model and Process Flow
- User Interface for presentation layer
- System Operation strategy
- Procedure

Business Architecture Strategy & Consideration

- Process/Data Integrity
- Streamline the Process Flow
- System Synchronization

Proposed Architecture Solution Strategy include:

Use Pattern to provide the strategy for each Business scenario, Logical process & Physical module. This pattern need be High-level, Full Process Strategy, Re-usable & Enterprise consideration

Full Logical Process Flow and Data mapping (between data and functionality)
High level impact and interfacing survey of the implementation scope within the system
Provide Development Strategy in Physical Implementation

- Process solution strategy
- Coding Language option
- Functionality solution strategy
- Integration strategy between each module
- Methodology (re-usable, CMM, RAD...)
- Coding guideline (review only)

Process Architecture Strategy (Logical)

Pattern 1.

Periodically Update and Multi-Steps (conventional)

1. Two steps commit, may cause Process and Data Inconsistency
2. May cause Data redundancy and process complexity
3. May waist system resource as it's performed by scheduler instead of event

Pattern 2.

Event Driven and Single Step (recommendation)

1. Two phase commit, ensure either (Both Commit) or (Both Rolling back)
2. Better performance as it's awake by event, and cursor is more efficient than others
3. Easier system maintenance as the code is reduced

Principles

The principles listed below are key guidelines for the design or purchase of applications and application components supporting distributed, client/server computing across the state.

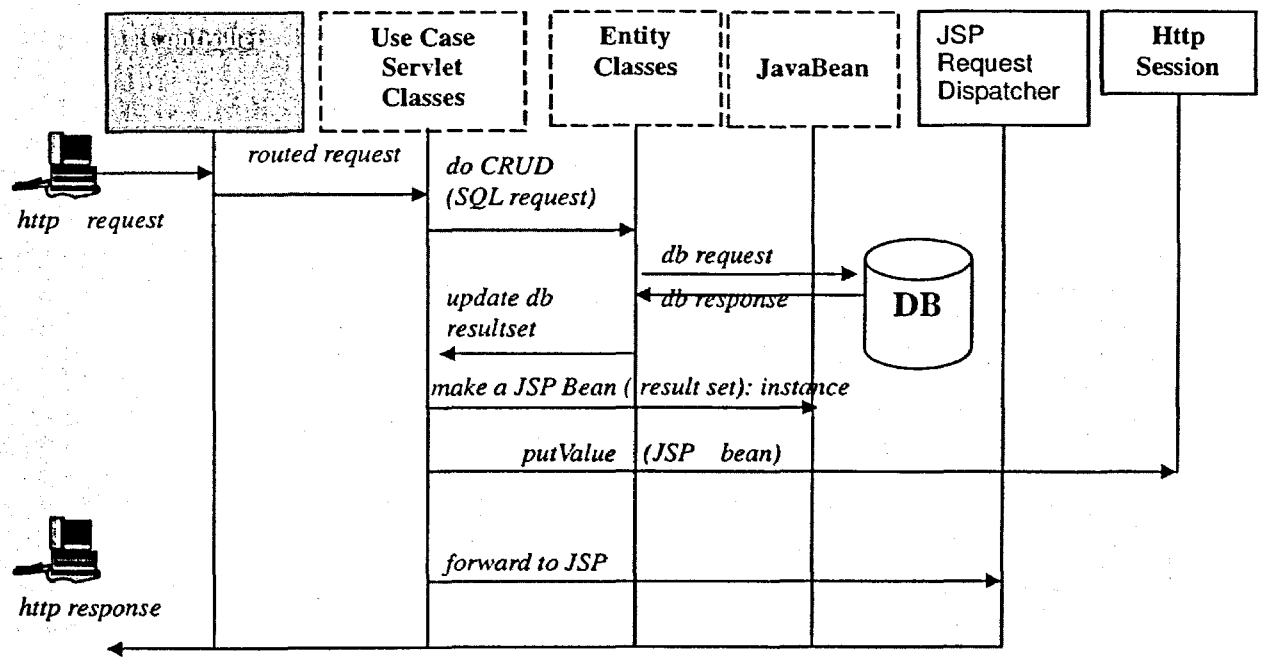
Principle 1
Design applications to be highly granular and loosely coupled
- . The designer should allow for the possibility of re-partitioning an application in the future.
- . Being highly granular and loosely coupled provide flexibility in physical implementation (i.e.,in the deployment of application components on different platforms).
- . Highly granular, reusable application components are key to increased productivity and rapid application deployment. The Componentware Architecture supports a highly granular design.

Principle 2
- Plan for extensibility and scalability
- . Applications must evolve to support new business requirements and make use of new technologies.
- . Extensibility provides functional scalability.

Principle 3
- Design application to reuse components
- . Applications should be built by assembling and integrating existing components, rather than by creating custom code. Shrinking cycle times do not allow for artisan programming.
- . Managing component reuse is supported by the Componentware Architecture.

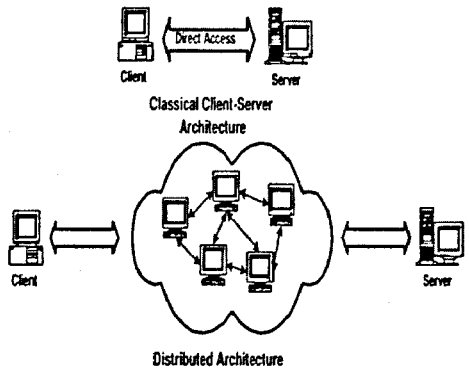
Thin Client

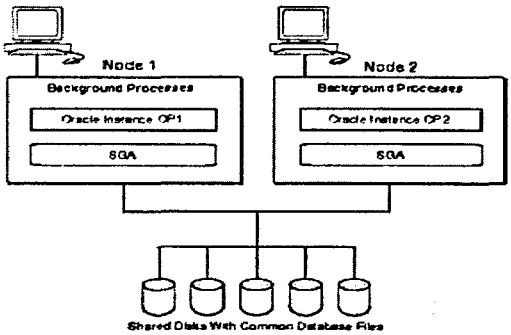
1. **The client (web browser) sends a CRUD (manipulation) request to the server via http;**
2. **The Servlet receives the request and creates a Record class and uses it for CRUID**
4. **The Record Class retrieves and returns any needed data to the servlet**
5. **The servlet performs processing on the DB data and creates a JavaBean containing the DB data**
6. **The servlet uses the JavaBean and a JSP to send the response to the client.**



Ref 5. Parallel Processing & Distributed System

Distributed System (1:N)





- ✧ **Distributed System : System carried out on a whole of computers without shared memory, sawn by the user like only one and single machine**
- ✧ **Dialogue between computers is allowed by a unique and global interprocessus communication mechanism**
- ✧ **Some processes may been executed in parallel on the network**

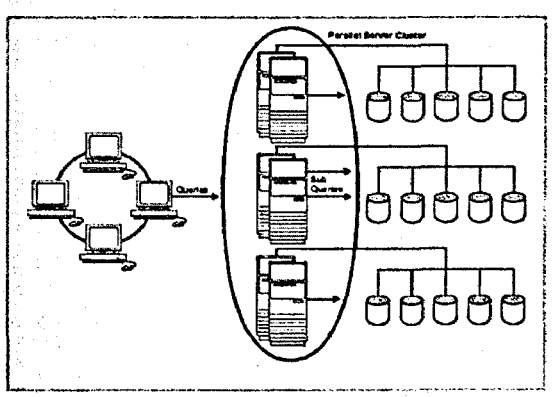
Parallel Server Instance Architecture (N:1)

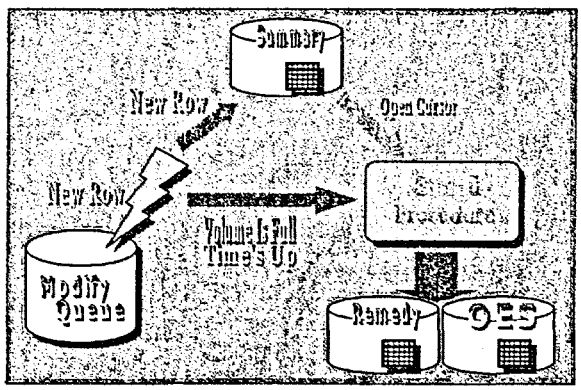
- ✧ Superior scalability through distribution of workload across nodes
- ✧ High-availability through multiple nodes accessing the database; if one node fails, the database is still accessible through surviving nodes

Parallel Query Operation (N:n)

Degree of Parallelism

SELECT /*+ PARALLEL <OBJ$, 2, 2> */ * from **OBJ$** where **COL$ = :COL_VAL$** ;
Query broken up n times for each node
(Parallel query statement) Query broken up between n nodes

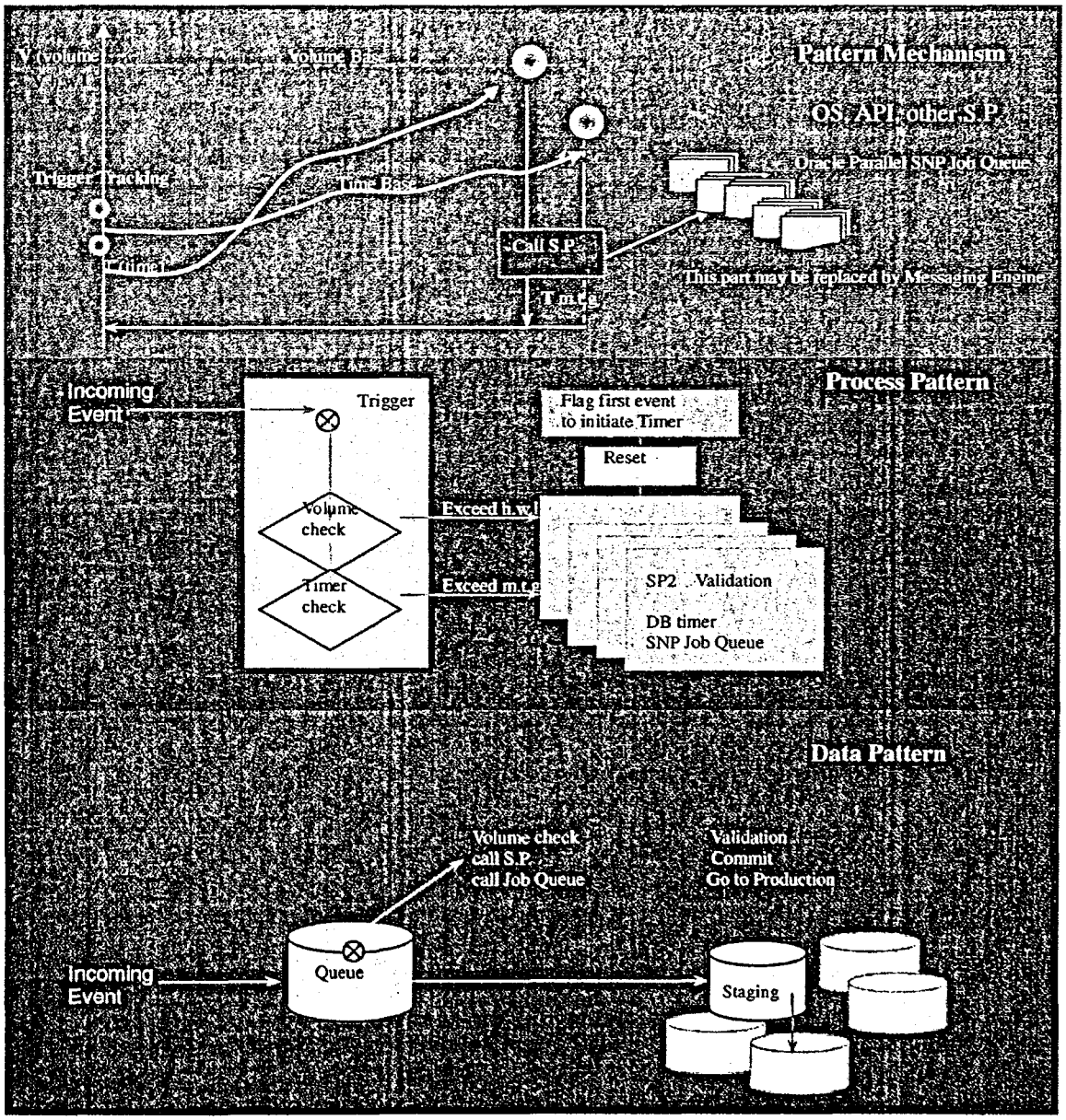




Event-driven Job Queue (Volume-Time engine)

Requirements:

- Process/Data Integrity
- Streamline the Process Flow
- *System Synchronization*
- *Volume & Time Control*
- *Event Driven (trigger, Oracle Package , PL/sql stored-procedure & function)*
- *Parallel Processing*
- *One-to-many centralized control*
- *Messaging Exchange Engine to handle the Job Queue*
- *Process step sequence and priority*



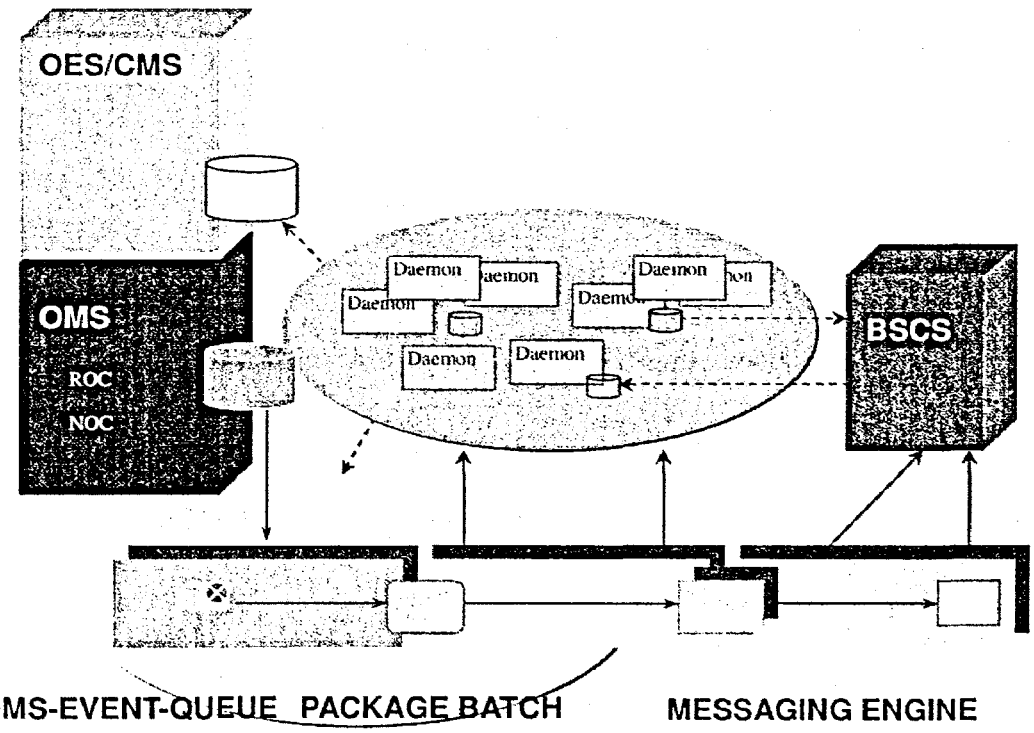
Recommended Technology Strategy

Event Driven (initially CMS-API-Validation, integrated-test stage, RA –auto. will be next, Ref. 1)
Trigger, Oracle Package, PI/SQL, Stored Procedure

Parallelism
DB Procedure Language (Ref.7)
Oracle Materialized View
Job Queue (Oracle Snapshot SNP background Process)
Distributed System (Ref.5)
Parallel Processing (Ref.5)

Job Queue Further Implementation

Process Architecture Strategy (Logical)
Periodically Update and Multi-Steps (conventional)
1. Two steps commit, may cause Process and Data Inconsistency
2. May cause Data redundancy and process complexity
3. May waist system resource as it's performed by scheduler instead of event
Event-driven Process Implementation



OMS-EVENT-QUEUE PACKAGE BATCH (stored-procedure, function)

- Event Driven
- Volume & Timing control
- Simple Logic control
- Call Oracle Package

- Cursor computation
- On-line for non-prioritized
- HTTP prioritized Chained jobs

MESSAGING ENGINE

- Message Sending & Receiving
- Queue Monitoring
- Priority Validation
- Queue job processing
- Routing

S10

Telecom Billing & Rating Architecture review

Vodafone, J-Phone Global Planning

vodafone

Raymond Wu

Enterprise Architect

Edition: 01
Version Draft 1.0

For Vodafone GITA Global Complioance

Approved by P. Thrush (GM)

Date: March 11, 2003

Operation

Contingency

The Contingency requirements in Billing system or some area in Customer Care can be higher than regular system. Automatics in Monitoring, Detection and Auto Switch-over may not required at the beginning, but Contingency plan such as; Primary Server (DB) and Secondary Server (DB) with Warm-standby and robust Switch-over mechanisms are definitely required in Billing and some area of the Customer Care systems.

The scenarios of switch-over is when primary server is down or encounter an outage during the batch process or need maintenance that cannot handle the user process and need the users be re-routed to the secondary server. When primary server become inactive, the secondary server acted as a new access point for possibly query-only service, all the non-query transaction need be stored in a staging area and wait until the primary server back to service, these stored transaction records then need be applied or sent to primary server after switch back.

Batch Process

Siebel/Portal Batch process mainly perform upload and download, combined with the daily batch process from OS level, the primary server need to switch the user access to secondary server. Batch process need an assessment for using Oracle partition or special Siebel/Portal configuration by using a code to flag the record for processing to eliminate the workload, so batch process run against only those small portion instead of a full-table-scan. Archiving by using Ju-box also need be considered for performance improvement.

The secondary database need be fully utilized, so during the server outage or heavy Batch process, the secondary server need to take over the access control and handle the user request until system switch the access control back to primary.

System Architecture & Integration

Siebel and Portal Package

Need to review those requirements that the Siebel or Portal packages cannot handle and need be customized. I would like to suggest the minimizing of such customization work to less than 10%. (this value stands 30% in the current Accenture's investigation)

Even Siebel said 70% (or higher) of standard Siebel solution is reasonable, I think at least 85% of standard Siebel solution is required. Please list and brief the out-of-Siebel-scope requirements so we can review them and decide the solutions. Example is the joining account case that Accenture raised in the meeting, the possible solution is; to apply and associate service sets from both family and organization to the account which need be beneficial from both sources, instead of changing the account hierarchical structure. *(please see diagram in ref.1)* The other possibility is applying profile to those account which need be impacted. Standard Siebel and Portal solutions doesn't guarantee the easier process or performance gaining in real-time & batch process, however I think we need to use these standard solution as much as we can, and tune the performance issues in each layer, this is one of the reason I am considering Parallel server to work on these issues.

Join layer between Siebel and Portal (UI, Services, Data layers...) need a strategy analysis for how to handle the common functionalities, examples are; Authentication, Authorization, Product/Account validation, sync. Either Siebel or Portal may handle these processes differently, which side need be activated as primary is a concern.

Data Architecture

Master DB

The pattern for Master DB decision is;
Master DB = Siebel DB + Portal DB + (a reasonable customized schema sets)
Regarding what portion can be the customized need a review and agreement.

According to Vodafone Global Architecture principles, both Siebel and Portal can co-exit in one database or separately exist in different database, but in either case data duplication is not allowed. The other requirement is Siebel standard Enterprise solution and database need be deployed as a priority;

Therefore, I agree Pattern 3 (customized database) should not be a long term plan and not need be assessed as we should use Siebel/Portal standard DB/Solutions as much as we can. The other point is the solution may become closer to pattern1 plus an
Enhancement feature, this feature provide DB contingency and provide solution toward performance issue during the Batch process.

This enhancement need to work together with the Contingency plan which include;

- Primary and Secondary warm-standby database architecture
- Oracle Parallel servers architecture is recommended for DB transparency
- Possibly Staging and Oracle partition solutions need a feasibility study
- Need to set the cut-off time for switch over, all the transactions happen after cut-off need be processed in the second day
- Need a robust switch mechanism (i.e.; use initORA files...) to switch user access between primary and secondary. So prim can switch the access control to secondary at cut-off, and switch back after prim complete the daily (batch) process
- Secondary DB may provide query-only services, may need a staging area (DB schema set) to take the transaction when primary become invalid, once prim complete the Batch job, the staging area sending the pending transaction to primary to roll the primary DB forward, system switch the access from secondary back to primary, and synchronize the prim to secondary DB by physical file copy.

(further information of design or research on these topics can be arrange)

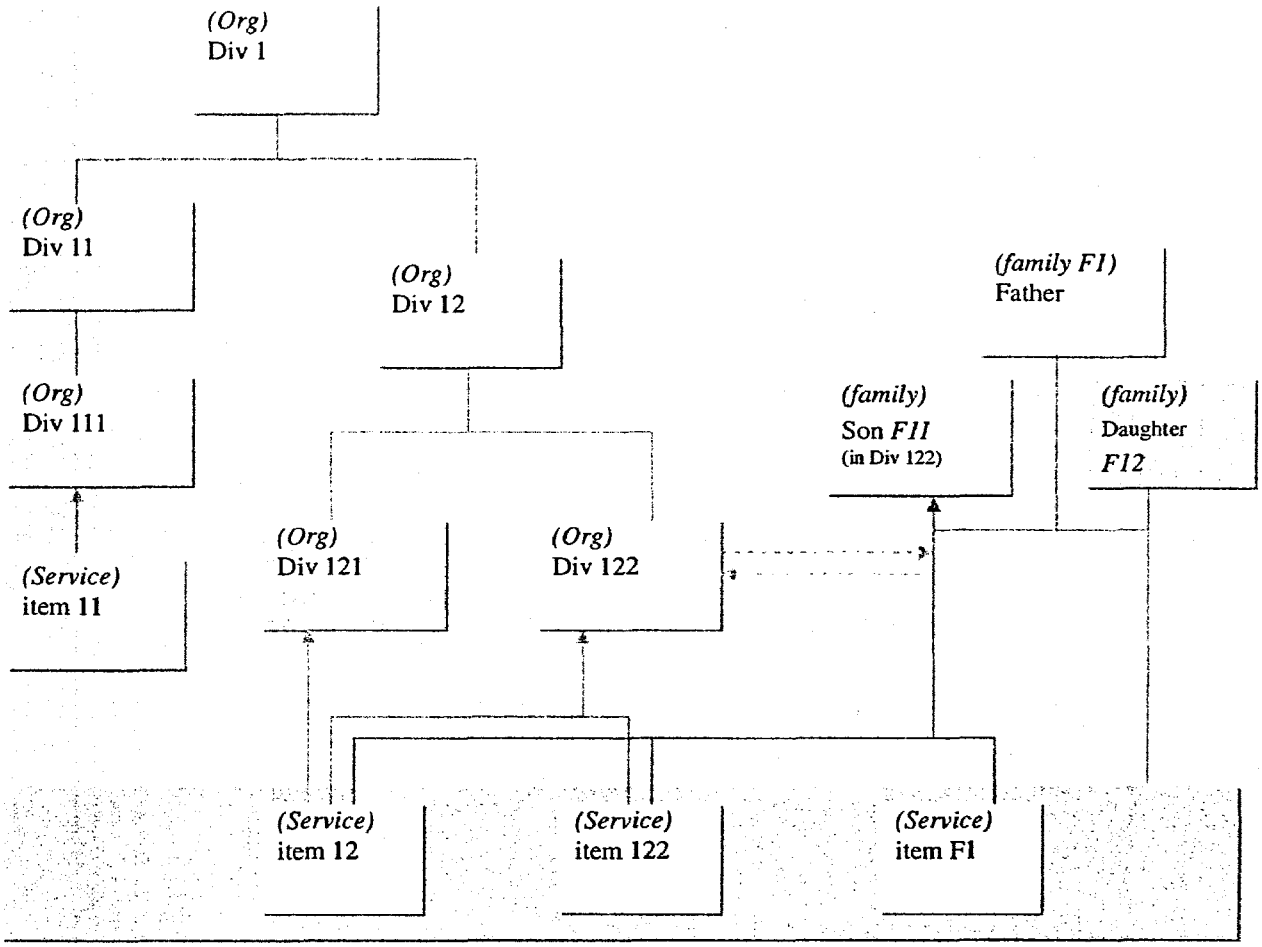
Infrastructure

Topology

Parallel servers is recommended to handle Siebel/Portal for both Contingency
and Performance purpose (refer Ref.2), this need to use Clustering and Distributed
technologies as the foundation. Oracle Parallel servers provide Oracle Instance on the server allow users to connect with a particular Oracle database as local database user, Oracle parallel server also provide clustering and distributed features that the Database servers are fully transparent with users.

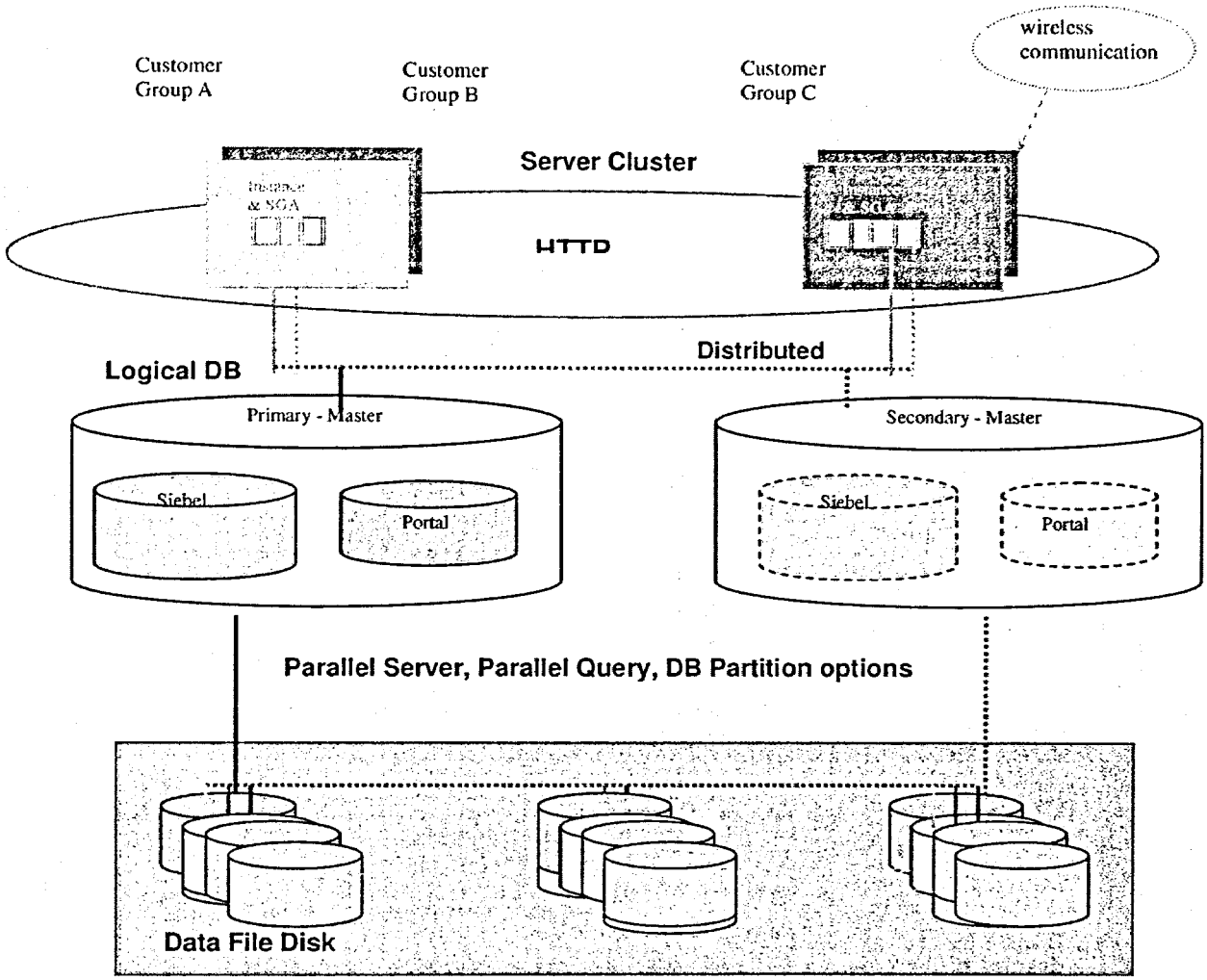
Oracle Cartridge also need be considered for capturing the business logic and processes associated with domain-specific data in user-defined datatypes. In some cases, where the data cartridge provides new behavior without needing new attributes, the vehicle of implementation may be packages rather than formal types. Once have defined these types using either of these approaches, Oracle determine the manner in which the server interprets, stores, retrieves, and indexes the data. Ultimately, data cartridges are the means to package this functionality as software components that can then be plugged into a server to extend its capabilities into a new domain.

A study case for joining multi-services toward an account



Assume son of family F1 work at Division 122, use "Service" to apply all services for Div. 122 and apply services for F1 be associated with the service account F11 (son)

A study case for Oracle Parallel server architecture



Currently VB program from MS-server front end takes the data change regarding Promotion and Rate plan to run against OES/CMS, and manually type the data and send it to remote to run the change against their database (CWS & OCR), the manual work starting from VB data input will be automated in this program that once administrator finish their data input, a click on a special button will generate the change from OES/CMS to remote format (CWS & OCR), that means the *"pr_gen"* engine use the schema mapping (between source and remote) to transform the data to remote DB format and create the commands to 'fly' the sql/shell script to remote and execute it.

The whole process need no effort at remote sites and if required, it can be upgraded to 'staging DB' in the future to allow the administer to validate the data input before the data goes to all (source and remote) sites. The only thing need to maintain from time to time is when the schema (at any site) is changed, *pr_mapping* need be run to re-generate the mapping. The synchronization mechanism use this mapping to consolidate the

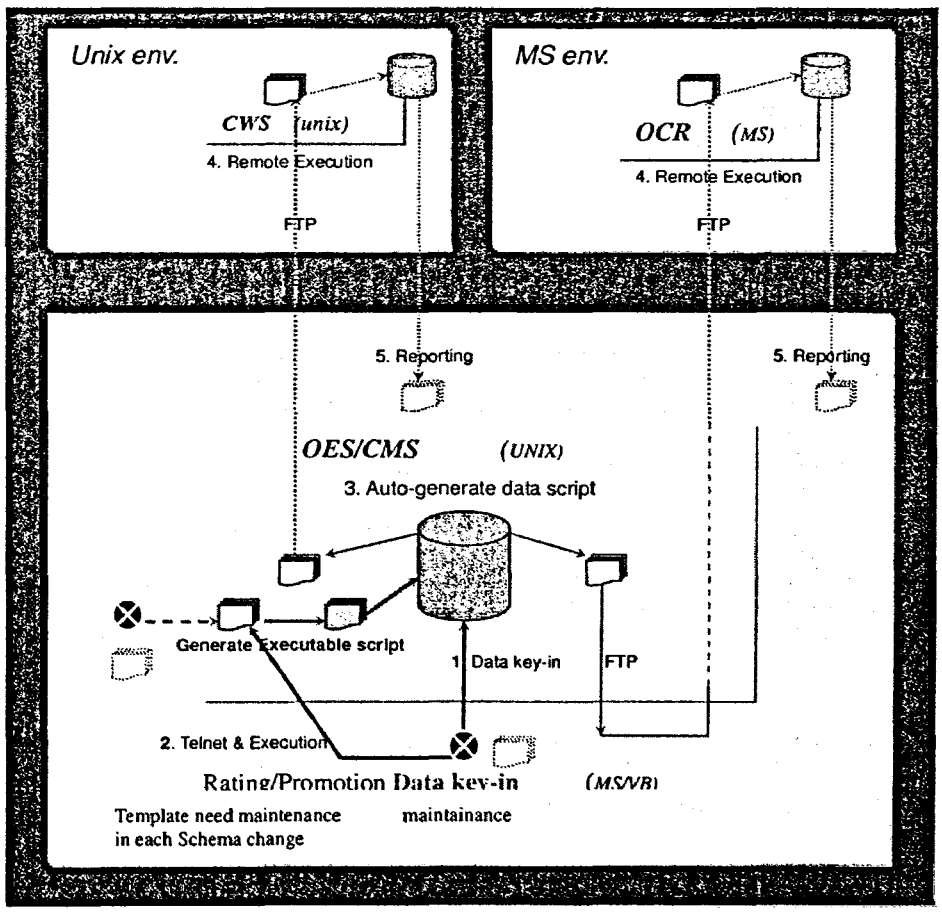
extracted data with the generated sql/shell scripts into *"pr_run_file"* and ftp it to the remote site, the pr_run_file has 2 different formats (Unix for Oracle and MS for SQL-server), depend on the parameter (ocr or cws) given, it will be run at remote site (*rsh* for Unix and *osql* for MS), the whole process takes around 5 seconds, however remote data change take no time from old data to new data.

This program is consisted of 2 major parts;
1. pr_gen (the engine to apply the change from source to remote), run "pr_gen ocr" (or cws) from VB
2. pr_mapping (generate mapping between source and remote schema) run "pr_mapping ocr" (or cws) from VB. The limitation of this program is, the data size better to be small (i.e.; less than 10k records)

Promotion/Rating Real-time Data Synchronization

(即時資料作業)

- 以 Broadcasting 方式進行一對多的 Data Synchronization, OES/CMS 為唯一允許變更資訊的 Data Source
- Sync 時機是在 OES/CMS 完成 Data key-in, 在最後按下 'Commit' 後以 FTP 及 Remote Shell Execution 方式一次完成,目的在避免重複 key-in 作業, 以及錯誤機率. Remote 端目前分 Unix (CWS) 及 MS (OCR) 二大環境
- 人工作業僅限 OES/CMS 上的 data key-in 及頻率極低的 schema change, 餘皆以 Shell/SQL 自動化處理



S11

Architecture Recommendation in Public Sector

Prepared by:

Raymond Wu
Consulting IT Architect
IBM Softwatre Group
ASEAN, Asia Pacific

Presentation to DBKL

5th June 2006

Objective

The new DBKL (Dewan Bandaraya Kuala Lumpur) lobby will be equipped with a fully integrated multi-channel payment and service delivery ICT initiatives. These initiatives together with the lobby renovation exercise will create a paradigm shift when dealing with DBKL - quality services with high tech high touch feelings, within an ambient and secured environment, and public satisfaction.

A crucial aspect of the enhancement of the DBKL lobby is the re-engineering of its customer service delivery channels. The face-to-face over the counter system will be revamped to allow DBKL to offer citizen facing services in a "mesra" environment, enabled by world-class technology.

The ICT application and integration at DBKL Lobby seeks to enable DBKL to achieve:

i) Presenting a unified, consistent, and branded image to the public

ii) Increasing the efficiency of services delivered to the public, measured by well-defined Key Performance Indicators (KPIs), for example, being able to handle a certain percentage (%) of public enquiries on the first contact

iii) Providing detailed and accurate public contact tracking and reporting capabilities

iv) Proactively look into steps to improve public service, by incorporating feedback mechanisms

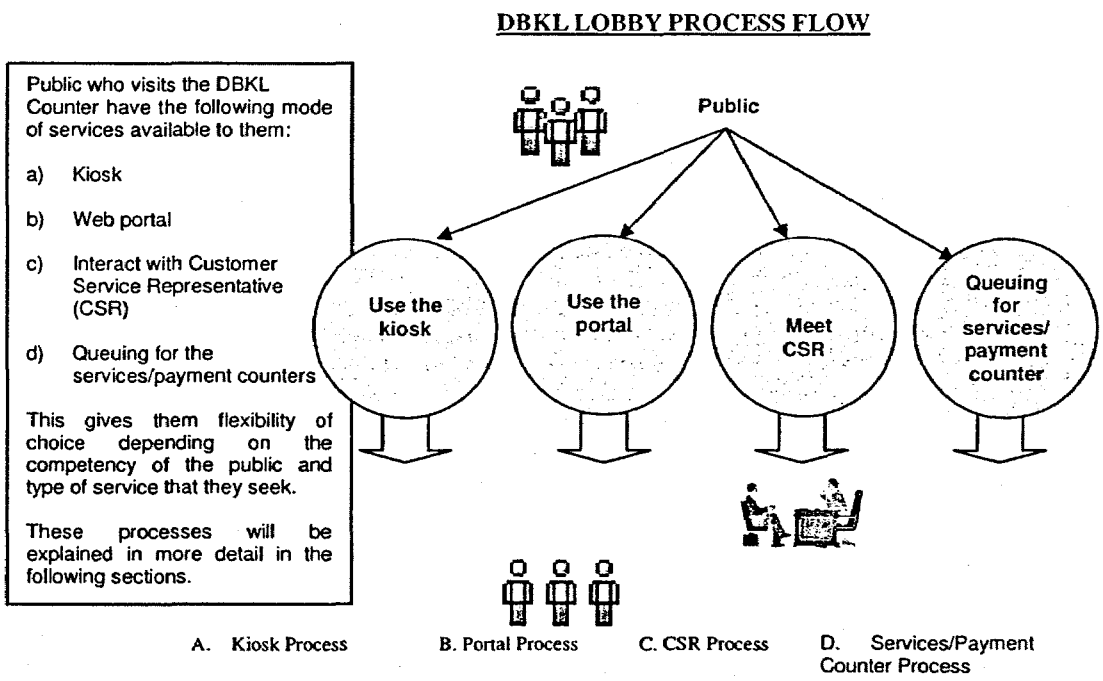
v) Providing anytime, anywhere and anyway access to DBKL services with high security measure

Major scope of DBKL MCP project

In general this ICT proposal covers the following five elements:
- System Integration (Front-end Gateway & Back-end Gateway)
- Customer Service Management (CSM) Application
- Payment Counter System
- Kiosk Application
- Online internet services/payment web portal application

System Integration cover;
- Front-end gateway
- Back-end gateway
- Customer Services Management Application Integration

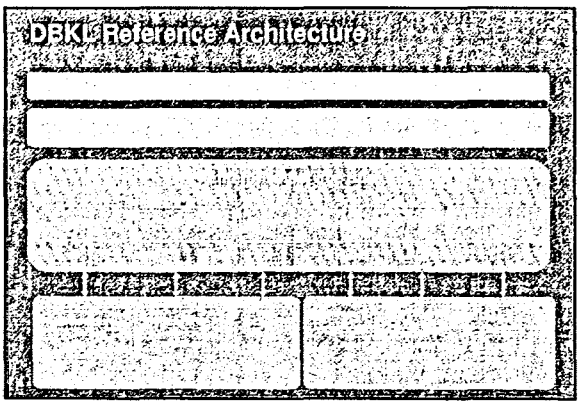
The following diagram provide a general guide into the process flow of public who visit the DBKL Lobby. These flows must be captured into the applications designed for the lobby.



The Big Picture

The new ICT application at DBKL Lobby will be an integrated system between multi-channels services and payment to the DBKL back-end systems available. This application will be called Customer Service Management (CSM) application. CSM will control the entire activities of services, payment and ICT management at DBKL Lobby. IBM has considered DBLK requirements carefully when applying its industry proven reference architectures.

The following IBM Reference Architecture proposed for DBKL takes advantage of IBM's leading Enterprise Gateway technology Enterprise Services Bus (ESB) to integrate the Core Business Process of Customer services CSM, Multi-channel platform MCP, Kiosk applications, Payment counter system, Web Portal application, on-line internet , 3rd party gateway adaptor (i.e.; SAP..) and provides an adaptive interface for existing system to be integrated into a fully transparent environment.



Impact Analysis of DBKL MCP project

The following are the features of the existing infrastructure :

- Integrated solution cover AIX, Linux and Windows environments
- Special RAID-5 implementation for disk installation & initialization
- Incorporating mainframe heritage LPAR parallel processing
- Capable of running multi-way processor and 32 bit to 64 bit computing
- Adaptability for major replication software to replicate data from Production site to Disaster Recovery site
- Multi-platform operating environment with capability to run the latest generation of OS/400, Linux, Microsoft Windows Server (via an IXA or IXS), AIX 5L, plus e-business application environments such as WebSphere and Java
- Extensive portfolio of proven solution available in all operating environments
- Enterprise-class dynamic logical partitioning for allocation and virtualization of resources – up to 10 partitions per processor
- Can support up to 1 to 1374 disk drives – 96TB of capacity (Existing, DBKL have 5TB)
- Up to 30 I/O expansion towers/drawers via High Speed Link
- Up to 443 PCI-X slots, 128 LANs
- Integrated Ethernet LAN and disk controllers
- Base 2-line WAN and base IOP
- Redundant, hot-plug components for additional reliability
- Rack-optimized design

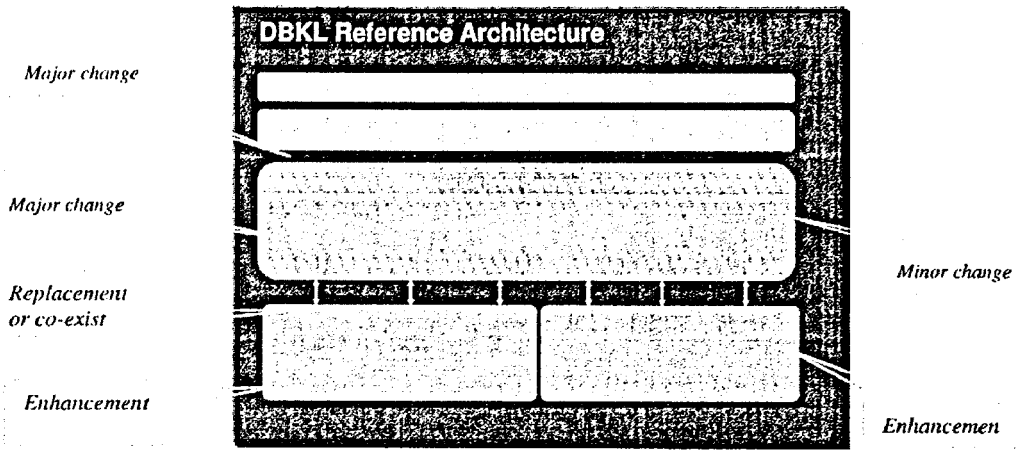
Currently, DBKL Lobby at Jalan Raja Laut provides 2 main counters, payment counter and information counter. The payment counters provide services for the public to settle their payment bills such as assessment bills, licensing, rental house bills and etc. The staff at the payment counters using the Receipting Machine (RM20) System which is based on DOS platform, standalone system and does not integrate with the back-end systems.

Conceptual Architecture from IBM solution

The investigation work mentioned above covers:

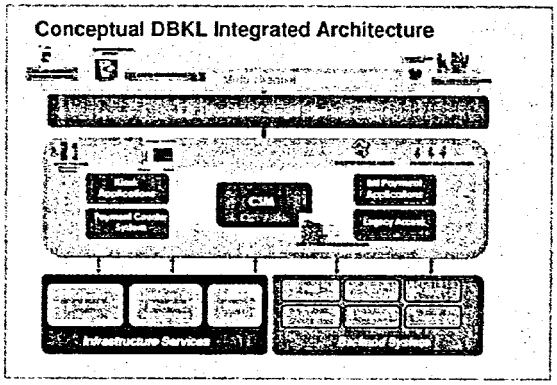
- Goal - IBM to provide solution to achieve DBKL's business objective
- Architecture - IBM reference architecture effectively achieves DBKL architectural and business requirements

Majority of the services system need integration with payment counter system.



- Detail requirements - Go through each key area and detailed functionalities to make sure these are covered
- Best Practice - IBM has used its vast array of best practices derived from successful implementations worldwide to create this solution.
- .We consider the main priority of DBKL is building up a common gateway for the major integration requirement in this project, IBM has been recognized the best practice in this enterprise gateway and integration. IBM is confident that the enterprise gateway (EBS) leading the IT industry, together with its maturity in areas that DBKL is looking for, we should be able to provide a smooth operation for DBKL to achieve its business goal.

Followed by IBM - DBKL reference architecture, the conceptual architecture below shows how the Multi-Channel, Common Security, Single sign-on and Portal layer leveraged in the system as User access layer, how the Enterprise gateway provides integration for the Core Business (CSM, Kiosk, Payment..) and how the infrastructure, security, monitoring, reporting and back-end system are integrated seamlessly into the Core business, front-end and back-end systems.



Critical Functionality

Integration

- Enterprise Gateway – The central gateway bring sall Front End, CSM and Backend together so the payment counter system can be accessed by using CSM application.
- Flexibility in rolling-out new Payment system - All the hardware at the payment counter system for RM20POS have options for either co-existence, migration or replaced by new PCS Payment System
- One Stop Gateway - Data received from one stop center agents shall be developed automatically integrate to the back-end using enterprise gateway. CSM should be working coherently and as a whole system solution.
- Total Systems Integration – Enterprise Gateway provide seamless integration

Integrity

- Single sign-on –differentiate user access rights, privileges and authentication
- One stop center agents data - extract and transmit non-DBKL transaction in softcopy

- Backend systems are fully integrated - The new payment system will be integrated with DBKL back-end systems through enterprise gateway, so the following back-end systems receive real-time data from PCS -
  - ✓ Financial Management System (FMS)
  - ✓ Pangkalan Data Undang-Undang (PADU)
  - ✓ E-License
  - ✓ Sistem Kawalan Pembangunan (SKP)
  - ✓ Others DBKL system will be define during user requirement study

Accessibility

- Interface for CSM Application – CSM application has access to PCS
- Real time updating - Real-time updating of payment data to DBKL back-end systems via front-end gateway, CSM and back-end gateway
- Online enquiry and validation - View and validate payment details from the DBKL back-end
- Offline collection - To work offline collection when system failure

Services

- Customer Service Representative (CSR) - Request from the public for the appointment with DBKL officers can be scheduled with Queuing tickets
- Kiosk – Provide self-service kiosk where public can interact with kiosk application for services and payment transaction.
- Procurement counter – counter for Bahagian Pengurusan Perolehan

Transaction Method

- Receive multiple and multi channels payment - To handle cash, cheque, credit cards, debit cards, ATM cards, Smart cards, Postal/Money order, etc..
- Used for multiple revenue codes - To cater for multiple types of revenue collection codes
- Barcode information capture - Capture transaction by barcode or any new technologies
- Payment collection – Both for DBKL revenue and one stop payment for non-DBKL bills. Other features such as bill endorsement, transaction storage, reporting & uploading data

Monitoring and Exception handling

- Audit Trail & System Log – Audit trail function for each transaction, modules, users etc.
- Generate reports - To consolidate revenue collection from multi-channels front end and generate reports based on User Requirement Study with DBKL
- Data Storage and Recovery - Perform back-up and restore function.
- Data consolidation - To consolidate all collection either from headquarters or branches
- Security and Backup Management – The coverage include security and backup system
- Archive data - System have function for archiving data.
- Receipt Cancellation - To handle cancellation of receipts
- Changing of Transaction Date - To change the transaction date for the supervisor level
- Analysis - The consolidated data from front-end reports will be sent to back-end system - Financial Management System (FMS) either automatically or manually on the scheduled collection.

IBM Solution profile and high level Architecture blue-print

- Portal and multi-channel accessibility
- Single Sign-on and LDAP
- Enterprise Gateway and Integration
- Kiosk and CSM adaptability
- Payment end-to-end solution

Method



M1 Enterprise Integration Strategy of Interoperability

IEEE-CISSE

M2 The determination of common services and aggregation service

IEEE-EISWT

M3 Service component design and automata theory

IEEE-EISWT

M4 Service Aggregation and Coarse Grained Based Design

IEEE-ICSSSM

M5 Decision Process in Component Based Design

ACM-NZ

M6 Componentization in SOA Development

EDS Corporation

M1

M1: Enterprise Integration Strategy of Interoperability

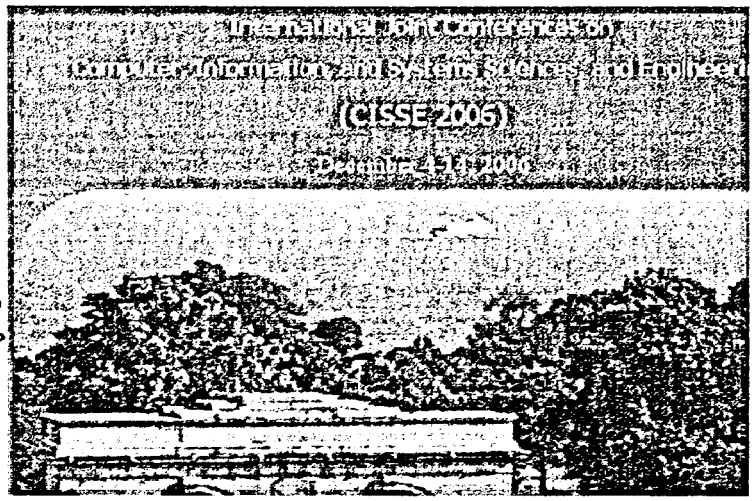
Publication: CISSE (IEEE)
ISBN: 978-1-4020-6263-6 (http://www.cisse2006online.org)

Publication Contact: Prof. Khaled M. lleithy
Assoc. Dean CS & Engineering
University of Bridgeport
elleithy@bridgeport.edu

IEEE (CIS²E 06)

Reviews for *Enterprise Integration Strategy of Interoperability*

Recommendation	Ratings		Notes
ACCEPT	Relevance	4	1. The author realizes that the development of enterprises must syncretize with IT and the IT field often lays behind the development of enterprises in the syncretism. The subject chosen by the author is constructive. 2. This article puts forward the architecture of enterprises integration under metadata environment. It can absorb new information effectively reduce the gap between IT and enterprise development.
	Originality	5	
	Soundness	3	
	References	4	
	Language	5	



IETA 06 Proceedings & CFP
TeNe 06 Proceedings & CFP
SCSS 06 Proceedings & CFP
EIAE 06 Proceedings & CFP
Conclusion
Program Committees
International

CIS²E 06

PROCEEDINGS OF THE
INTERNATIONAL CONFERENCE ON SYSTEMS, COMPUTING SCIENCES AND SOFTWARE ENGINE
(SCSS 06)

[illegible]

[illegible] [illegible] [PowerPoint Presentation] [Multimedia Presentation]

[illegible]

[illegible] [illegible] [illegible]

[illegible]

[illegible] [illegible] [PowerPoint Presentation]

[illegible]

[illegible] [illegible] [PowerPoint Presentation] [Multimedia Presentation]

[illegible]

[illegible] [illegible] [PowerPoint Presentation] [Multimedia Presentation]

[illegible]

[illegible] [illegible] [PowerPoint Presentation] [Multimedia Presentation]

57 Semantic Representation of User's Mutual Trust Model — Raymond Wu, Jie Liu — IBM and UCS University of Glasgow

[illegible] [illegible] [PowerPoint Presentation] [Multimedia Presentation]

Enterprise Integration Strategy of Interoperability

Raymond, Cheng-Yi Wu
IBM Software Group
P.O.Box 3077, 7 CBP1, Singapore 486048
Faculty of Information Technology
University of Technology, Sydney
P.O. Box 123, Broadway, NSW 2007, Australia

Jie Lu
Faculty of Information Technology
University of Technology, Sydney
P.O. Box 123
Broadway, NSW 2007, Australia

Abstract

In this new computing age of high complexity, a common weakness in the interoperability between business and IT leaves IT far behind the direction business is taking; poor business responsiveness and IT governance makes it even harder to achieve the enterprise goal. To cope with this common issue, we introduce the enterprise interoperability to integrate the metadata between business, service and information layers, this create visibility of vertical alignment within enterprise architecture and use metadata configuration to construct the mapping between each layer.

Introduction

The main purpose of this paper is to conduct a research in the enterprise architecture issue as a means of addressing the vertical gap with new knowledge contributions in the vertical integration of the interconnectivity of cross-layers include direction, business, process, service and information.

We identify this overall tactical issue that emits from the vertical gap as the "invisible enterprise issue" and these are the findings in our early analysis stage. Realization require multi-dimensional aspects from business innovation, service strategy, solution architecture, implementation and integration [1].

Two key processes were used to achieve this objective. The first process relates to how the enterprise goal may be achieved by impacting business process and its service components level, which we call "business semantic", the reason componentization needed to be set first because visibility and traceability [2]. The second process follows one level below the first one and is called "information metadata." It constitutes the repository of metadata storing information and its process of moving information between service and data; it thus supports the upper layers by repository synchronization. The micro process integration was initiated by the domain context analysis process. H. Lee & J. Lee pointed out, once sets of use case scenarios are formulated, the next step will be the extraction of verbs and nouns for identifying domain objects [3]. The three different types of repositories (metadata, services and semantics) collaborate each other. the fact that all of the organizations share the same service means they are all managed under the same rules which were developed using the top-down approach [4], [5].

Discussion

First Area - Technical Environment Setup
First, the concept of "centric master data and single view of information" to consolidate all fragmentation into master data which called "single source of truth". Before the built, the metadata was categorized into two groups - system and transaction, the transaction metadata was related to those business transaction while system metadata was related to those system configuration

ISBN: 978-1-4020-6263-6

and DDL (data definition language) which supported parameterization in metadata computing. Metadata computing was followed by the built of metadata repository, the reason of the introduction of recursive metadata iteration and cursor computing was an intention to build the metadata engine in communicating with other repositories in Java/XML/SQL formats [6]. One of the examples was by allocating summarized "course" information in metadata repository as driving force in extracting associated students' study plan gained the query response time from several seconds down to one nanosecond.

Second Area - Business Process Strategy

There are two steps to achieve "componentization", as we were all aware that traditional silo implementation aimed end-to-end business process, it supported project architecture but also caused many problems of fragmentation in enterprise architecture. The concept of componentization takes the identified business component(s) into decomposition process and produce common elements, the process use litmus test for its boundary decision, when we take the componentization concept into realization, the key of this phase was using DCRP reuse methodology [7] in achieving this goal and was the important creative work in this phase. The DCRP reuse methodology was intended to address the vertical integration concerns which were currently labor-intensive development and maintenance issue inherent in multiple instances of the enterprise architecture. Interoperability capability in a vertical direction allows mediation computing entities such as messaging carrier and class mapping (i.e. XML and JDBC) choreography between service and information to become easier and leading toward a seamless vertical integration [8]. The integration need an interlingua as mediation solution, the constraints and dependency of interoperability between two applications and tendency that the integration has been moving downward as the high complexity combination of application, process and database integration enforce the mapping between theories, and these require separate mediation in integration.

Third Area - Enterprise Integration

The realization of enterprise Integration initiatives started from the outcomes of previous two areas in deriving a "loosely coupled" mediation (which leveraged to service oriented architecture's principles) in creating configurable linkages between business and IT. In this final stage, the resolutions and approaches were leveraged to those common methodologies and architecture practices (such as Zachman, RUP, SOA and MDA, to proceed the verification and validation of the approaches we had. This is evident from the nature of business direction, virtualization and interoperability perspectives [9]. *Desmond DSouza define vertical integration in his* "Model-Driven Architecture and Integration Opportunities and Challenges" said, vertical Integration came from different levels of abstraction of the same system, from physical data models, logical data models, networks, applications specifications, component assemblies, business process models, and business goals and strategies [10].

Enterprise Interaction Strategy

The semantic interpretation from business to service, and from service to technical need decomposition process to work on those selected business components, so each business component can be decomposed into business use cases for further breakdown. Service components derived from business use case are the mediation between business and technical element which integrate all business semantic, interfaces and technical configuration into a logical component, initially all the service components are in atomic level [11] [12]. S. Melnik (1999) proposed a universal interface which avoids common models and languages, the Generic

ISBN: 978-1-4020-6263-6

Interoperability Framework developed to facilitate integration of heterogeneous information systems. [13]. Based on this common framework, we can build the component of both functional and non-functional by using industry standard. DSouza specified the standard and common component as followed - To manage models of large scale, a coherent modeling architecture is required, with standard ways to describe shared concepts, rules, patterns, frameworks, mappings, and generators. [14]. M. Schmalz argued that database schema and application code can be efficiently derived from various types of schema representations, particularly the relational model, and supports comparison of a wide variety of schema and code constructs [15]. According to L. Yu et al., An obstacle to software reuse is the large number of major modifications that frequently have to be made as a consequence of dependencies within the reused software components. Component base implementation is the foundation in coping with this bottleneck [16]. Therefore the determination of Common Services and Composite Services is a use cases driven parsing process, the purpose of the parsing iteration process is trying to find out the common areas and dependency between services. The determination of common component and service aggregation, as mentioned, is through the parsing process using use case driven model.

C. Foste et al. indicated Metadata has a key role to play in allowing systems to be built and automated. A management system composed of multiple distributed components implements a service such as content validation, transcoding, search or generation; and relies on metadata information to achieve success [17].

In building up the enterprise interoperability between layers, componentization will be the foundation for integrating business and service components to derive the architecture of common component and service aggregation, store the intelligence of mapping and routing information in repository in metadata form, this strengthen the foundation of enterprise vertical integration. However, the tactical issue in the ontology skeleton is that currently there are no principles to direct each layer when working on the above-mentioned three issues using vertical integration. In other words, when we are looking for the principle of "separation of concerns" in vertical integration, to maximize the output from equation –

$$\tilde{O}_{output} = \sum \hat{O}_n = \sum (d_n \times b_n) \times (p_n \times s_n) \times i_n$$

Each $\hat{O}_n$ needs to be a concatenation from an assembly of "atomic" level components; each component can be the assembly of a semantic sub-context (d_n), a business sub-component (b_n), a decomposed sub-process (p_n), a fine-grained service component (s_n), or a piece of metadata (i_n). The routing of $\hat{O}_n$ forms a map for approaching a business context. Before approaching the resolution, we need to have an insight into the ontology principles in order to determine the framework. The reason the SOA approach is required in this foundation and should be extended to the business layer is because we need to create a broad "loosely-coupled" and "separation of concern" ontology environment covering all business, process and service layers.

Ontology skeleton in Integration strategy

Referring to Figure 1 – the concept of the ontology approach is based on the driving force of the business layer which takes the business context and translates it into a semantic message. The semantic message stored in the repository further directs the decomposition and input of the element message to the service component which we call "business semantic injection." Referring to Figure 2 - the business component in each cell denotes a unique and independent business entity with an individual function. Figure 3 – what we are looking for is a driving force in creating a

ISBN: 978-1-4020-6263-6

common foundation of "semantic injection" downward to a service component so that each service component holds an associated business semantic element from micro process. For an overview of how the business context and semantic message can be transmitted to the component, please refer to Figures 4, 5 and 6. Figure 4 shows that the business component and its element decomposition are accomplished by means of a parallel process; one is "logical" and is a class object element while the other is a "physical" component which carries the message.

Metadata Strategy

The database partition shown in Figure 7 is an efficient way to achieve a "separation of concern." By storing the partition information in a metadata repository with linkage to service metadata, the system integrity and performance will be improved dramatically. Each pending record can be centralized in 1% of the database for further service until completed. By this means each query hit-ratio is nearly 100% and greatly improves SLA (service level agreement) efficiency. Thus interoperability can be attained within the meta-layer to impact the individual entity effectively [18]. In each strategic implementation, the interoperability is built by using semantic metadata between each layers. Figure 8 shows how to top down and bottom up approaches, how the information and service form common components and aggregated components in creating the vertical coherence of enterprise integration.

Conclusion

The concept of "enterprise vertical integration" covered many aspects of technology and knowledge which we already discussed, the issue was from misalignment of business and IT, so the business context and semantics cannot effectively impact IT development and cause all development, change and maintenance very difficult. The lack of interconnectivity between context, semantics, service and data is the tactical issue with the clear implication that enterprise architecture needs a robust micro-mechanism of semantic messaging and metadata to coordinate across layers. The alignment can reuse the mapping between business service and information component from those successful experience, however in most of the project scenario, only part of the empirical transformation process from business to service and to physical element can be reused, and leave a large portion of the alignment require starting from business decomposition. In our current research, an insight in how to start from business component to derive the service component will be described in a separate research paper.

ISBN: 978-1-4020-6263-6

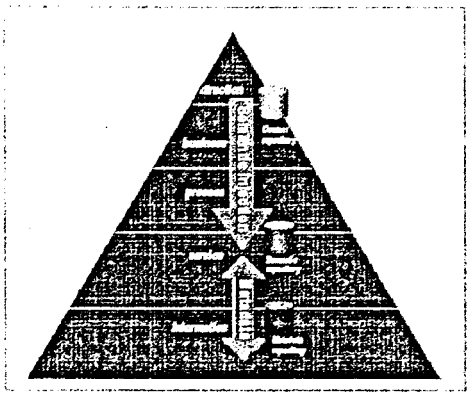


Figure 1 – Conceptual ontology interoperability between layers by using "semantic injection" and "metadata"

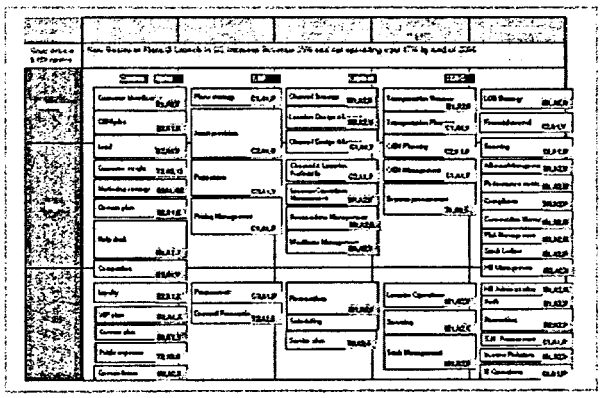


Figure 2 - Mapping between context and semantic notation

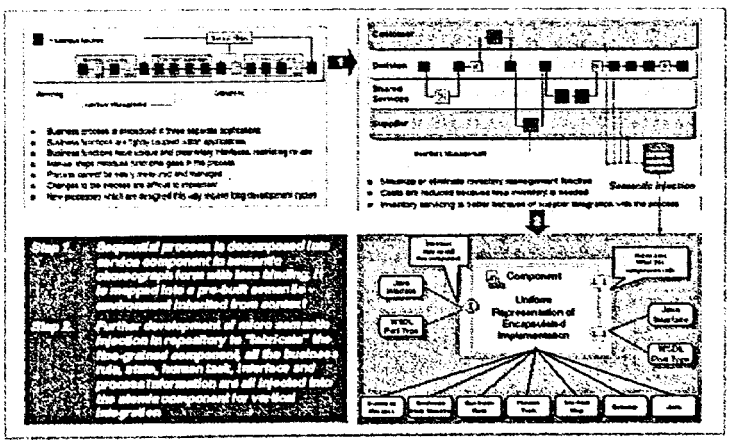


Figure 3 – Through business component modeling, each business element is identified as a strategic direction element to provision a specific business goal

ISBN: 978-1-4020-6263-6

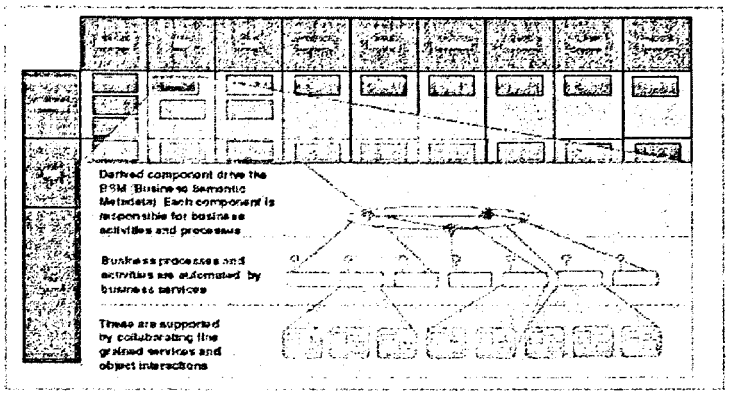


Figure 4 – Business semantic injection into object model

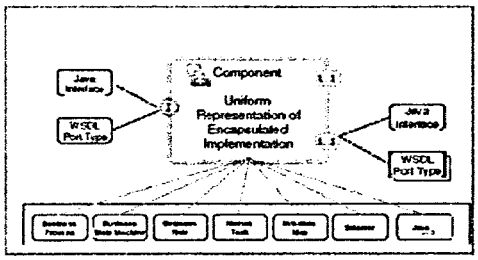


Figure 5 – Service component used as a dynamic micro repository holding business semantic

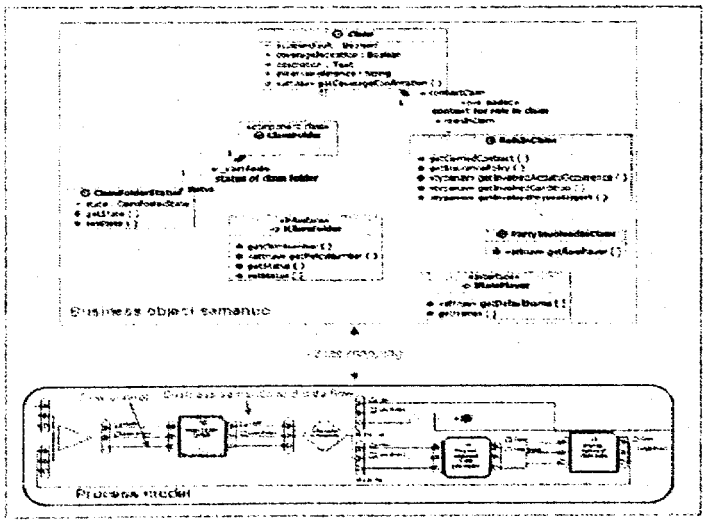


Figure 6 – Service component inherited semantic micro properties and assembly into process by using business rules

ISBN: 978-1-4020-6263-6

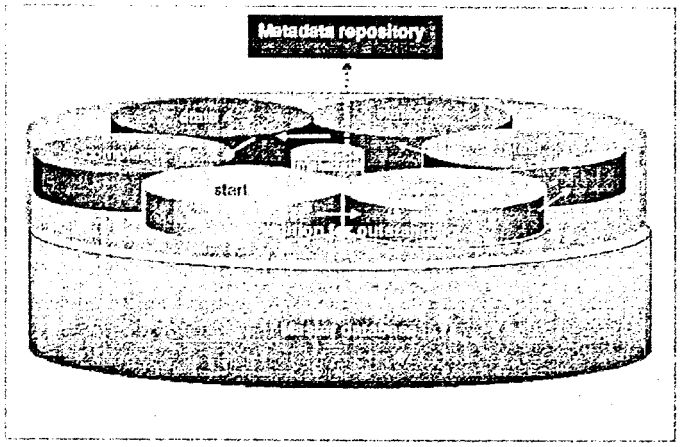


Figure 7 - Database partition in metadata environment

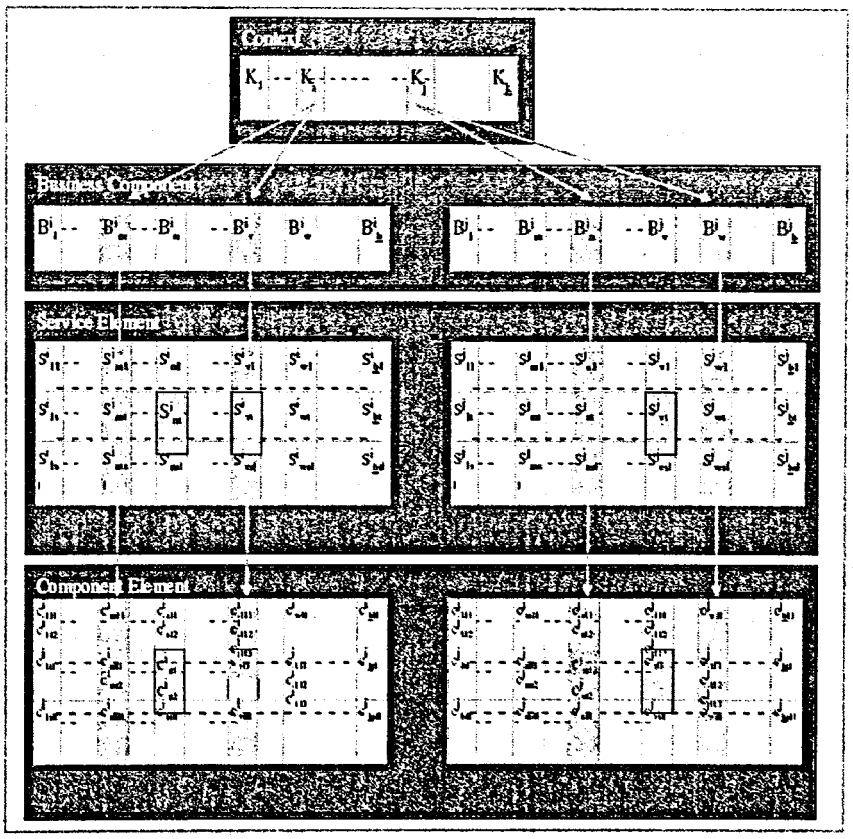


Figure 8 - Metadata in component element provisioning the component services for upper layers of services and business

ISBN: 978-1-4020-6263-6

Reference

1. W. Yao, (2005), "Interoperability: an indispensable propellant of e-China", International Journal of Services and Standards 2005 - Vol. 1, No.4 pp. 446 – 452
2. M. Lytras, (2006), "The Semantic Electronic Government: knowledge management for citizen relationship and new assessment scenarios", Electronic Government, an International Journal 2006 - Vol. 3, No.1 pp. 5 – 17
3. Heeseok Lee and Jae Lee (2006), "Analyzing Business Domain: A Methodology and Repository System", Korea Advanced Institute of Science and Technology
4. C. Wagner, K. Cheung, R. Ip, S. Bottcher, (2006), "Building Semantic Webs for e-government with Wiki technology", Electronic Government, an International Journal 2006 - Vol. 3, No.1 pp. 36 - 55
5. G. Moynihan, R. Batson, (2006), "Development of a matrix methodology for database reengineering and improvement: an e-government case study", Electronic Government, an International Journal 2006 - Vol. 3, No.2 pp. 190 - 203
6. Tannenbaum, Metadata Solutions: (2001), "Using metamodels, repositories, xml, and enterprise portals to generate information on demand", Addison Wesley Professional, Boston, MA
7. Dublin Core, (1997), "Dublin Core Metadata Element Set", Version 1.1: Reference Description, http://purl.org/dc/documents/rec-dces-19990702.htm
8. C. Liu (2005) "Using e-governmental indicators to build virtual value chain", Electronic Government, an International Journal 2005 - Vol. 2, No.3 pp. 277 – 291
9. Batini, R. Grosso, G. Longobardi, (2006), "Design of repositories of conceptual schemas for large-scale e-government projects", Electronic Government, an International Journal 2006 Volume 3 - Issue 3 pp 306 - 328
10. ntti Vehviläinen, Olli Alm and Eero Hyvönen: (2006). "Combining Case-Based Reasoning and Semantic Indexing in a Question-Answer Service", June 20, 2006. Poster paper, 1st Asian Semantic Web Conference
11. M. Staab, N. Stojanovic, R. Studer, and Y. Sure, (2001), "Semantic Portal - The SEAL approach. to appear in: Creating the Semantic Web.", MIT Press
12. O. Valkeapää and E. Hyvönen, (2006), "Semantic Annotation with Browser-based Annotation Tool", 1st Asian Semantic Web Conference
13. Sergey Melnik, (1999), "Generic Interoperability Framework", Distributed and Parallel Databases Journal 1999
14. DSouza,(2001)," Model-Driven Architecture and Integration Opportunities and Challenges", Kinetium
15. M. Schmalz, 2003, "EITH – A Unifying Representation for Database Schema and Application Code in Enterprise Knowledge Extraction" , 22nd International Conference on Conceptual Modeling
16. L. Yu, (2005), "Categorization of Common Coupling in Kernel-Based Software," 43rd ACM Southeast Conference, 2005
17. C. Foster, P. Stentiford, (1999), "Metadata - The Key to Content Management Services, Proceedings of the Meta-Data", The third IEEE Meta-Data conference

ISBN: 978-1-4020-6263-6

M2

M2: The Determination of Common Services and Aggregation Service

Publication: EISWT/ IEEE CS Proceeding (http://www.promoteresearch.org/2007/eiswt/index.html)

International Conference on Enterprise Information Systems and Web Technologies (EISWT-07)

July 9-12, 2007 Orlando, FL, 32819-8284, USA.

Publication Contact: Prof. John E. Edward

Editor in chief, EISWT

Montana State University

jeedward@gmail.com



"John Edward" <jeedward@gmail.com> Add to Address Book Add M

From: Yahoo! DomainKeys has confirmed that this message was sent by gmail.com more

To: rwu_academic@yahoo.com

Subject: EISWT-07

Date: Fri, 19 Jan 2007 18:51:51 -0500

Dear Colleague(s)

On behalf of the committee of the 2007 International Conference on Ent Information Systems and Web Technologies (EISWT-07), I am pleased t you that your paper: The determination of common services and aggrega service

Raymond, Cheng-Yi Wu
IBM Germany
Faculty of Information Technology
University of Technology, Sydney
P.O. Box 123, Broadway, NSW 2007, Australia

ID:eiswt105
has been accepted for presentation at the EISWT-07 conference to b Orlando, FL, USA during July 9-12 2007. Congratulations! The paper be included in the proceedings. The reviewers' comments are attach email. The conference organizers should receive the revised and formatted final paper, copyright form and the registration fee on or bef 2nd 2007, in order to include your paper in the conference proceedings.

Sincerely

John Edward
(on behalf of EISWT-07 committee)

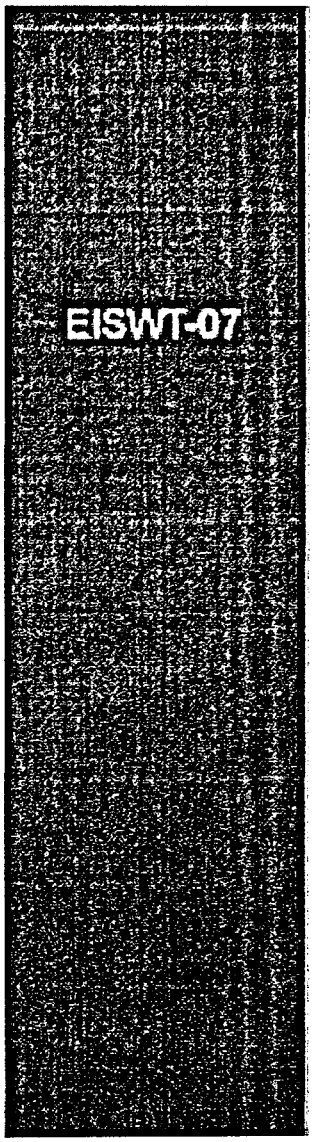
Acceptance/rejection will be based on the scores below. 1=worst, 5=bes

Relevance to conference (1-5): 5

Originality (1-5):4

Contribution (1-5):4

Technical quality (1-5): 4



PROCEEDINGS OF THE 2007 INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS AND WEB TECHNOLOGIES

Editors:
Bobby Granville, N.S.Kutti,
Michele Missikoff and
Ngoc Thanh Nguyen

Associate Editors:
Scott Graham, Naphtali D. Rishe
and Booserm Wongsaroj

Orlando, Florida, USA
July 9-12 2007

CISSE'07

ISBN: 978-0-9727412-4-8

Contents

SESSION: Databases and XML
SESSION CO-CHAIRS: Scott Graham, Naphtali D. Rishe and Booseem Wongsaroj

The Determination of Common Services and Aggregation Service 36
Raymond Wu

The determination of common services and aggregation services

Raymond Wu
IBM Software Group
raymond@de.ibm.com

Abstract

Common service and aggregation services are determined by a rational model rather than by a political decision. We applied parsing techniques in use case simulation and dependency analysis to produce the common services and aggregation services.

Our goal is to provide visibility for those common components which are frequently deployed and of high dependency, and which aggregate fine-grained to coarse-grained services. These components construct within the enterprise a skeleton which is business-critical with high implementation priority. The skeleton exploits a loosely coupled approach in creating integration interoperability and constructing the foundation for service virtualization.

Introduction

Vertical enterprise integration is a metadata-driven approach in the component-based environment. There are six steps in its implementation and realization: 1) validation of business components; 2) validation of service components; 3) validation of physical components; 4) common component parsing analysis; 5) service aggregation through parsing analysis; and 6) determining implementation priority and skeleton within enterprise integration. Our focus in this research paper is Steps 4 and 5, as we need to determine the candidacy for common service and service aggregation [1].

Both political and rational models are constructed in a component-based environment; this is because a conventional development environment provides no opportunity of group decision-making or rational analysis. With all the contributing factors in a scrambled code, any changes or add-ons can have a huge impact. Even the scope of the impact area and components can be difficult to determine, thus the impact is high while the interoperability is low. In conventional development, weak visibility between business, service and information makes the linkage and enterprise interoperability impossible. Once the alignment is missing or invisible between layers, the common services and service virtualization become impossible as these require a foundation in both macro interoperability and micro mechanism. This results in a silo system with serious limitation of services and weakness in enterprise integration [2].

"Common service" means a service component such as has often been used in the use case driven process. In the customer-based service industry, an "account status query" can be easily identified in many requests or processes. Once we can align this common service to both physical and business components, we call it a common component. The concept of common component can encapsulate both conceptual and physical elements in provisioning a service. A common component is not limited to a functional element; any information or system driven element such as business object, portal UI element, data access protocol or component interface would qualify as a common component candidate. Common service is a conceptual term with alignment to both business components and physical objects; these can be either functional or non-functional elements. An "account balance query" may consist of a combination of "multi-account identified" functional components and "database query" non-functional components. The service aggregation parsing process identifies these two services as candidates for common service; they are validated once the frequency passes the appropriate threshold and their interconnectivity design is seen as feasible.

Service aggregation means a composite service aggregated from at least two components that follow a certain sequence and perform a combination of coarse-grained services. Service aggregation is validated when the parsing statistics show a combination of services following a unique sequence frequently forming either an aggregated service or a potential service-based process; the pre-built coarse-grained service provides upward service provisioning and brings in its values in eliminating the integration interface and risk caused by boundary implementation [3]. Service metadata holds mapping and routing intelligence that is stored in a service repository so that common components and service aggregation have an opportunity to form the main route of the common service skeleton; any business semantic translation to the process and infrastructure level becomes feasible and efficient under this main route skeleton [4].

Once the visibility of vertical enterprise alignment is created, the metadata of a component-based configuration forms a clear picture of the enterprise skeleton using metadata integration between business, service and information. Through a metadata approach, we can either reuse the metadata mapping and configuration from successful experience or construct a new map for the project by using the configuration of common components and service aggregation as a project portfolio while extending the implementation to configuration of less significant

components. Moreover, once the service components and aggregation services are built they can either be reused or aggregated to a coarser grain later if the aggregation is common and optimized in the parsing evaluation. These processes produce metadata interoperability for enterprise integration.

Literature review

DSouza specified the standard and common components as follows: "To manage models of large scale, a coherent modeling architecture is required, with standard ways to describe shared concepts, rules, patterns, frameworks, mappings, and generators." For example, the development of different applications with object models mapped to relational tables requires a single object-to-relational mapping architectural style from a shared model as a refinement framework. If all the enterprise models maintain a separation of information, engineering, and computational views which are then integrated with them in particular ways, this common structure should itself be modeled in a shared architectural style model as a package pattern [5].

According to L. Yu et al., an obstacle to software reuse is the large number of major modifications that frequently have to be made as a consequence of dependencies within the reused software components. Component-based implementation is the foundation for coping with this bottleneck [6]. Therefore, the determination of common services and composite services is a use case driven parsing process; the purpose of the parsing iteration process is to discover the common areas and dependency between services. The determination of common components and service aggregation, as mentioned above, is through the parsing process by using a use case driven model.

J. Hunter argued that metadata interoperability is a fundamental requirement for access to information within networked knowledge organization systems. For this purpose, a common underlying data model has been developed to enable the scalable mapping of metadata descriptions across domains and media types. This model provides a set of basic building blocks for metadata modeling and recognizes the importance of "events" to unambiguously describe metadata for objects with a complex history. To test and evaluate the interoperability capabilities of this model, some real multimedia examples have been applied to analyze the results of mapping from this data model to a variety of different metadata domains. To overcome the limitations in semantic mapping, the metadata "thesaurus," capable of providing additional semantic knowledge, has been developed to enable flexible and dynamic mapping among metadata standards [7].

The methodologies we used in this research work are: AHP (analytical hierarchy process) in four levels of derivation including RMA (rational model analysis); CCSA (common component and service aggregation) process; K-B-S-C (KPI, Business, Service and Component) transformation; semantic metadata; and interoperability with reference to the definition given by D. Wang and T. Koch. We will first describe the concepts of semantic metadata and interoperability by using the definition from D. Wang and T. Koch.

Parsing techniques in component analysis

Before we proceed to the key area rational model, we use the architecture terminology "K-B-S-C transformation" to represent interoperability between architecture layers. We begin with enterprise strategy (K), the chief area of concern for stakeholders and business managers, to derive the components in both business (B) and services (S) layers. As each layer of K (context), B (conceptual) and S (logical) is at the conceptual or logical level, the bottom layer which is the physical component (C) forms a loosely coupled alignment with S in provisioning bi-directional interoperability. The micro mechanism between these two layers forms the service unit; aggregate individual service provides service in the business layer to finally achieve the strategic goal.

The denotation is summarized as follows:

K: KPI - strategic Key Performance Identifier and all concerns of the stakeholder

B: Business Component - unique business block - each block may perform individual and independent business

S: Service element - logical service component at an atomic level

C: Component element - physical component associated with a service component; a business object is a typical example

Once the validation of business components and their associated service components is completed, we can move on to service provisioning based on the components we selected. The provisioning is an architectural process to produce both common and composite components by means of use case parsing to count the frequency and dependency of service components. Frequency means a count of the appearance of a component in the parsing process; dependency means that two or more components work together frequently and follow a particular sequence in forming a composite service or process.

In the use case parsing process, first the finite service matrix S_{ij} (i = 1, 2, 3 ... n, j = 1,2,3 ... n) is plotted in both vertical and horizontal dimensions in order to predict both the common degree and the degree of sequential dependencies. We have designated the map produced by the plotting a CCSA (common component and service aggregation) map; the process is called a CCSA process. Each service S, when plotted in the CCSA map, refers to the aggregation of service aggregation. This means that a combination of service components, if they follow a certain sequence with a certain frequency, may qualify for service aggregation; that is, two or more atomic service components can be aggregated upward to a coarse-grained service in which the grain size

and threshold is determined by the architecture group using either empirical benchmarking or optimization in the evaluation of advantage (i.e., coarse grain helps in minimizing interfaces) and disadvantage (loss of flexibility in the new configuration).

In component evaluation, for their significance in relation to common degree and level of aggregation, let S* denote the finite set of services,

$S^* = \{ S_1, S_2, S_3, \ldots S_n \}$, $n \geq 2$.

Let S^{cc} denote the finite set of services which qualified as a common component, we derive $S^{cc} \subseteq S^*$, let $\widetilde{V}_k^*$:

being the normalized index showing the intensity of common degree

For kth service, k = 1,2... n, let $\mathcal{V}_{threshold}$ be the threshold in qualifying a common component, the ratio of $\widetilde{V}_k^*$: over $\mathcal{V}_{threshold}$ determines whether S_k is qualified as a common component.

In the evaluation of the significance factor of service component in K-B, B-S and S-C transformation, let

$P(X^k_j)$ be the jth member's weight for kth KPI

$\Pi(X^k_{ji})$ be the jth member's preference of component X_i for kth KPI

$\Phi(X^k_i)$ be the generic significance factor which is a concatenation between the weight of the role and preference of component X_i for kth KPI

$\Phi^*(S^{kb}_i)$ be the business significance factor in the B (business) dimension, which is a concatenation between the weight of role and preference of component S_i for kth K (KPI) and bth B (Business component)

$\Phi^*(C^{kb}_i)$ be the physical component significance factor in the I (information component) dimension, which is a concatenation between the weight of role and preference of component C_i for kth K (KPI) and bth B (Business component)

K-B transformation (size of Business component = n, size of decision group = m)

$\Phi(B^k_b) = \Sigma P(B^k_j)\ \Pi(B^k_{ji})$, let $\mu B^k = Max(B^k_b)$, denote the maximum value for B^k_b

$_{1<b<n\ \ 1<j<m}$

$\Phi^*(B^k_b) = \frac{\Phi(B^k_b)}{\mu B^k}$ be the relative value for $B^k_{b\,(1<i<n)}$ when compared with μB^k

B-S transformation (size of Service component = n, size of decision group = m)

$\Phi(S^k_i) = \Sigma P(S^k_j)\ \Pi(S^k_{ji})$, let $\mu S^k = Max(S^k_i)$, denote the maximum value for S^k_i

$_{1<i<n,\ 1<j<m}$

$\Phi^*(S^{kb}_i) = \frac{\Phi(S^{kb}_i)}{\mu S^k}$ $\Phi^*(B^k_b)$ be the relative value for S^k_i when compared with μS^k

S-C transformation (size of Component element = n, size of decision group = m)

$\Phi(C^k_i) = \Sigma P(C^k_j)\ \Pi(C^k_{ji})$, let $\mu C^k = Max(C^k_i)$, denote the maximum value for C^k_i

$_{1<i<n,\ 1<j<m}$

$\Phi^*(C^k_i) = \frac{\Phi(C^k_i)}{\mu C^k}$ $\Phi^*(B^k_b)$ be the relative value for C^k_i when compared with μC^k

S-S transformation (CCSA parsing analysis)

Let $\Psi(S^k_{ij})_{(1<i<n,\ 1<j<n)}$ be the matrix from the result of n-dimension S-S parsing

Let $\gamma(S^k_{ij})$ denote the count of S^k_{ij} (dependency count from S^k_j to S^k_i)

When i = j, $\gamma(S^k_{ij})$ is the statistic of common degree of S^k_{ij}

When i< >j, $\gamma(S^k_{ij})$ is the statistic of dependency degree of S^k_j on S^k_i

$$\Psi(S^k_{ij})_{(1<i<n,\ 1<j<n)} = \begin{bmatrix} \gamma(S^k_{11}), \gamma(S^k_{12}), \ldots \gamma(S^k_{1n}) \\ \gamma(S^k_{21}), \gamma(S^k_{22}), \ldots \gamma(S^k_{2n}) \\ \cdots\cdots\cdots\cdots \\ \gamma(S^k_{n1}), \gamma(S^k_{n2}), \ldots \gamma(S^k_{nn}) \end{bmatrix}$$

Since $\Phi^*(S^k_i)_{(1<i<n)}$ represents the significant factor of the service component using the political model, and the concatenation of K-B & B-S

$\Phi^*(C^k_i)_{(1<i<n)}$ represents the significant factor of the service component using a political model, and the concatenation of K-B & S-C.

This implies that the significance of service component S depends on both its associated business component and itself, as does component element C. The reason we use a parsing matrix to represent the significance factor is to try and derive dimensional vectors by means of different patterns.

Both $\Phi^*(S^k_i)$ and $\Phi^*(C^k_i)$ statistics are shown in Figure 1 as an example of CCSA parsing analysis associated with atomic service components in provisioning upward service through the alignment of S-C, S-S and B-S transformation.

Aggregation in the service layer is important as it eliminates boundary configuration, increases the integrity of common services in an error-free configuration environment and, furthermore, can be developed into coarse-grain processes or processes adjacent to the business layer. Therefore the referential alignment of B-S strategy uses coarse grains to support business components directly; those fine grains at the bottom of the service layer can form a different pattern in supporting the coarse grains. This implies that the fine grains

serving as the foundation of the services can be reused so that one service can contribute to provisioning many coarse grains through different patterns. The formation of a pattern can use common components as a base as they are common in provisioning generic services in each business component. Aggregation that is based on a pattern derived from parsing analysis as a usual case is crucial in common services; in the terminology our use of "common" indicates an encapsulated enterprise service that is seldom changed even though a light revision is acceptable.

The information aggregation in the component layer forms the metadata that is stored in an information repository distinct from the service components. It constructs part of the component element in the physical layer. The component element consists of physical units of functional and non-functional elements; while an agent is an example of a functional element, a business object or a security component are both considered as non-functional elements.

According to C. Foste et al., metadata has a key role to play in allowing systems to be built and automated. A management system composed of multiple distributed components implements a service such as content validation, transcoding, search or generation, and relies on metadata information to achieve success. Metadata specification, security, transport and storage needs have been identified through the implementation of prototype components. Although the technology employed in each of the management components is often straightforward, the impact on organizational processes and interworking may be significant [8].

Service aggregation is a process for deriving the metadata within a layer, so even though the service components and component elements have been well defined in Steps 1 through 5, fine tuning is required to derive the final step.

In Step 4 we described how the service component is aggregated. Within the component layer there is no such process; as the physical layer is at an atomic level loosely coupled with peer components and associated services, we assume that all the component elements in information component form are in an optimized state and that no further aggregation is required. Figure 2 illustrates the concept of how the data is aggregated toward the service layer, with several steps being required to streamline the data transformation from raw data to abstract information. Therefore, before any information component is defined, a suitable aggregation in optimization is necessary; this implies that the aligned service component is always associated with an optimized component. Thus, if the component is an information component, there is no further aggregation required after S-C is configured.

In each K implementation, the interoperability is built by using semantic business deconstruction to derive service elements and component elements; at this point each element can be aggregated and composed into composite components or identified as a common service for reuse and shared service purposes. There are three layers of assessment in determining the tendency of each service component.

- Business layer - group decision to determine which business component to use
- Service layer - group decision with some analysis work from architecture
- Component layer - group decision with some analysis work: an information pre-built foundation in meta form is required to make sure the information metadata is well integrated in this layer
- Service transformation – identify services in relation to common services and service aggregation

Priority in the political and the rational model

Let $\Phi^*(SC^k_i)$ denote $(\Phi^*(S^k_i) + \Phi^*(C^k_i))_{1 < i < n}$, and represent the vertical intensity of service component i (under strategy k) in both upward business and downward information dimensions; $\Phi^*(SC^k_i)$ is actually a matrix produced by the political model. Meanwhile, we introduce $\Psi(S^k_{ij})$ to denote the matrix of counts from our use case parser

$$\text{Let } \Psi(S^k_{ij})_{(1<i<n,\,1<j<n)} = \begin{bmatrix} \gamma(S^k_{11}), \gamma(S^k_{12}), \ldots, \gamma(S^k_{1n}) \\ \gamma(S^k_{21}), \gamma(S^k_{22}), \ldots, \gamma(S^k_{2n}) \\ \cdots\cdots\cdots\cdots \\ \gamma(S^k_{n1}), \gamma(S^k_{n2}), \ldots, \gamma(S^k_{nn}) \end{bmatrix}$$

Eigenvalues approach

Assume the value we derived from the political model $\Phi^*(SC^k_i)$ satisfies the eigenvalues approach and works as an eigenvalue of the rational model.

In $n \times n$ matrix, the rational model approach of S-S in deriving common component and aggregation from the matrix of $\Psi(S^k_{ij})_{(1<i<n,\,1<j<n)}$

In the equation $\Psi(S^k_{ij})\ X = \Phi^*(SC^k_i)\ X$, X denotes the eigenvector set; each set can be a pattern of common service. The implementation priority is based on the intensity of component reusability and frequency of fixed pattern. There is a possibility we may be able to use the above equation in deriving the patterns; however, how to best use a factor in balancing the left-hand side of the equation which concerns rational model parsing and the right-hand-side political model derivation is a concern which needs further development.

Based on the analysis of M. Selvage et al., once we have the priority of component implementation ready the roadmap is decided; at this point there are several different implementation patterns to achieve semantic interoperability:

- Point-to-point semantic integration
- Hub-and-spoke semantic integration
- Master data management (MDM) pattern
- Industry information model
- The semantic web

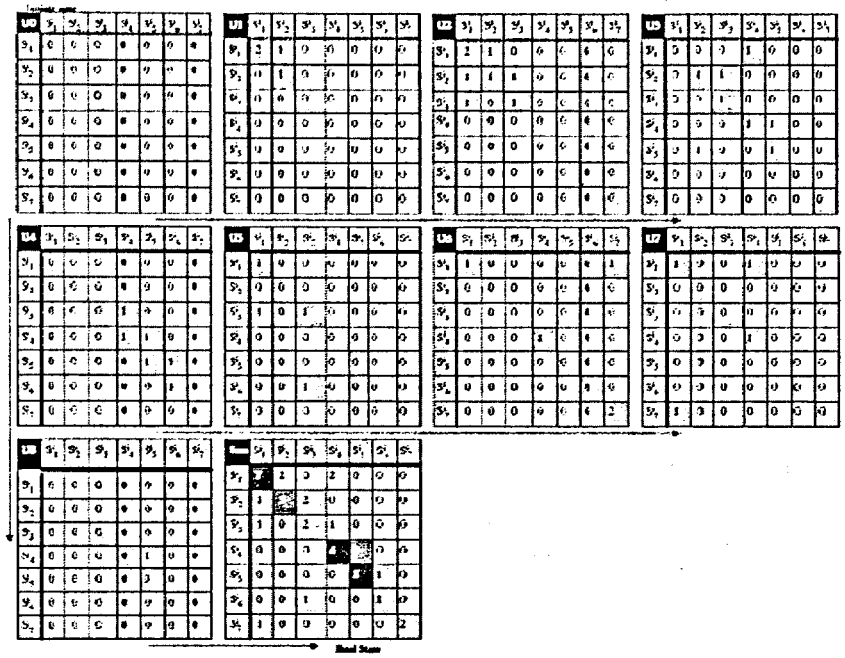


Figure 1. CCSA parsing analysis

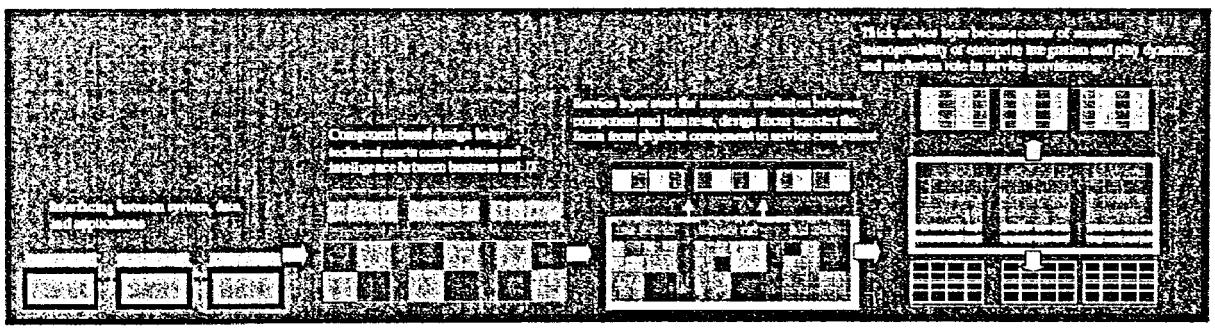


Figure 2. Transformation in constructing common component and common service

The scope can include business units, enterprises and groups of enterprises within the same industry or across industries. The bigger the circle of semantic interoperability is, the more reusable the results will be in the long run but also the more difficulty there will be in the areas of coordinating and reaching consensus. Moreover, an individual company will have less control over tailoring the reusable assets to its unique business needs and determining the desirable features. For these very reasons, some companies would use internal proprietary data models for their internal integration efforts, but use industry standard models for B2B. Enterprises should take a balanced view to consider the tradeoffs of adopting different patterns of semantic interoperability [9].

Conclusion

Each project or enterprise activity has its unique strategy, business context or KPI that first needs to be converted to business components. The alignment can reuse previous experience in a top-down K-B-S-C approach by selecting the same business components, service components and physical components to establish the business goal; however, in most project scenarios only part of the empirical K-B-S-C transformation process can be reused. This leaves a large portion of the alignment as either new material or a deep modification which requires starting from business component selection.

In the service-component-based architecture, interoperability within K-B-S-C is the key in building the skeleton of enterprise integration in a vertical direction. To cope with this objective, we apply both a political model and a rational model to validate the components in each layer. The political model applies group decision-making while the rational model applies parsing analysis; both models work on component validation with different focus of scope.

In role-based group decision-making, each role has a different weight in each layer within B-S-C and each group member has a different preference in their component selection. We use the political model, starting with the business components. We repeat the political model in a use case deconstruction to derive service components and physical elements. The group members validate both service components and component elements by using the same method with differently weighted roles. The evaluation between service components and physical components is based on how significant their contribution to the business goal is from a logical and physical information component perspective; a higher degree of information aggregation of C within B-S-C gives a higher preference weight in that physical component. The multiplication of B and S significant factors represents the significance of S within the K-B-S-C transformation; using the same method, the multiplication of B and C significant factors represents the significance of C within the transformation. The objective of K-B-S-C transformation and significance factor evaluation is to try to build up the vertical enterprise skeleton and extend the key services to their peripheral surroundings in performing full service virtualization.

In the following K-B-S-C political model evaluation, a bottom-up approach using rational model parsing is applied to S-S transformation. This step is the last one in K-B-S-C transformation and the objective is to derive common components and service aggregation. The statistics of parsing results construct a matrix of common components and aggregated coarse-grained components which can be a candidate for process; this bottom-up approach attempts to create visibility of common services in either component units, interface or composite components with their priority and dependency. This provides the architects with a roadmap for building up their enterprise integration and strengthening the alignment and interoperability within the K-B-S-C transformation.

Reference

1. Raymond Wu, (2006), "Enterprise Vertical Integration in the e-Government", Electronic Government, an International Journal (EG)
2. S. Dwarkadas K. Gharachorloo L. Kontothanassis D. Scales M. L. Scott R. Stets. (1998) ," Comparative Evaluation of Fine- and Coarse-Grain Software distributed Shared Memory", Cambridge Research Laboratory
3. Curtis, K., Foster, P. W., Stentiford, F. (1999), "Metadata - The Key to Content Management Services", The third IEEE Meta-Data conference, April 6-7, 1999, Bethesa, Maryland, USA
4. DSouza.(2001)," Model-Driven Architecture and Integration Opportunities and Challenges", Kinetium
5. L. Yu, (2005), "Categorization of Common Coupling in Kernel-Based Software," 43rd ACM Southeast Conference, 2005
6. J. Hunter , (2001), "MetaNet - A Metadata Term Thesaurus to Enable Semantic Interoperability Between Metadata Domains", Journal of Digital Information, Volume 1 Issue 8
7. C. Foster, P. Stentiford, (1999), "Metadata - The Key to Content Management Services, Proceedings of the Meta-Data", The third IEEE Meta-Data conference
8. M. Selvage, D. Wolfson, B. Zurek, E. Kahan, (2006), "Achieve semantic interoperability in a SOA", IBM Academy 2006

M3

M3 Service component design and automata theory

Publication: EISWT/ IEEE CS Proceeding (http://www.promoteresearch.org/2007/eiswt/index.html)
International Conference on Enterprise Information Systems and Web Technologies (EISWT-07)
July 9-12, 2007 Orlando, FL, 32819-8284, USA.
Publication Contact: Prof. John E. Edward
Editor in chief, EISWT
Montana State University
jeedward@gmail.com



From:	"John Edward" <jeedward@gmail.com> Add to Address Book Add M Yahoo! DomainKeys has confirmed that this message was sent by gmail.com more
To:	rwu_academic@yahoo.com
Subject:	EISWT-07
Date:	Sat, 20 Jan 2007 18:06:56 -0500

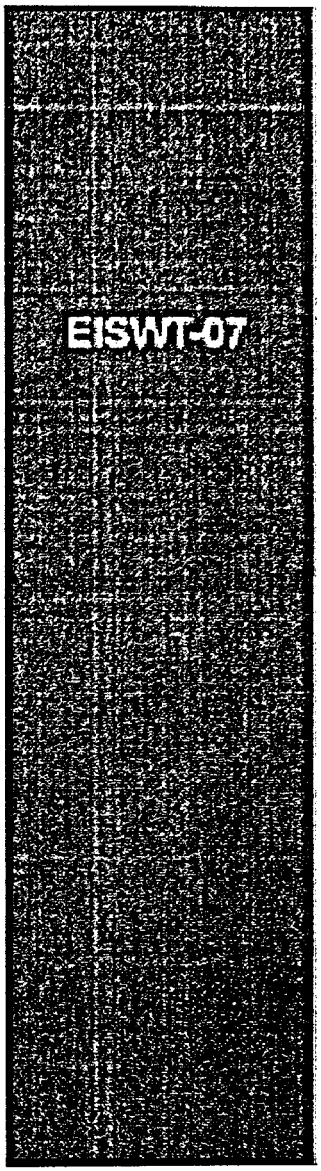
Dear Colleague(s)

On behalf of the committee of the 2007 International Conference on Ent
Information Systems and Web Technologies (EISWT-07), I am pleased t
you that your paper: Service component design and automata theory
Raymond Wu
IBM Germany
Faculty of Information Technology
University of Technology, Sydney
P.O. Box 123 Broadway
NSW 2007, Australia

ID: eiswt108
has been accepted for presentation at the EISWT-07 conference to b
Orlando, FL, USA during July 9-12 2007. Congratulations! The paper
be included in the proceedings. The reviewers' comments are attach
email. The conference organizers should receive the revised and
formatted final paper, copyright form and the registration fee on or bef
2nd 2007, in order to include your paper in the conference proceedings.

Recommendation: xa) Full paper b) Short paper c) Reject

Review comments for communications to authors:
The idea is convincing and interesting.



PROCEEDINGS OF THE 2007 INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS AND WEB TECHNOLOGIES

Editors:
Bobby Granville, N.S.Kutti,
Michele Missikoff and
Ngoc Thanh Nguyen

Associate Editors:
Scott Graham, Naphtali D. Rishe
and Booserm Wongsaroj

Orlando, Florida, USA
July 9-12 2007

©CSREA 2007

ISBN: 978-1-60132-412-4-6

Contents

SESSION: Databases and XML
SESSION CO-CHAIRS: Scott Graham, Naphtali D. Rishe and Booseram Wongsaroj

Service Component Design and Automata Theory 53
Raymond Wu

2007 International Conference on Enterprise Information Systems and Web Technologies (EISWT-07)

Service component design and automata theory

Raymond Wu
IBM Software Group
raymond@de.ibm.com

Abstract

Decision, analysis and design are key steps in building the skeleton of enterprise integration. In this research paper we deal with the component interoperability design stage by introducing the aggregator and automata concept as the foundation of service component design, applying Automata Turing Machine theory in developing the component evaluation to construct the aggregation services. This work is based on the common component concept we implemented in previous research and on the component-based design foundation we build in the current research. We further explore the micro mechanism of component interference and service provisioning in achieving service virtualization.

Introduction

In the current research work, we shift our focus to the design stage to examine how the components are designed as atomic units, how they work together for interconnectivity and how they achieve the enterprise goal in vertical interoperability. As the architecture design and configuration is defined in the analysis stage, our focus of design is the micro processes of component manufacturing, interconnectivity between components and micro process design of service provisioning; this is accomplished in a macro service environment which covers service aggregation and dimensional provisioning for a business goal [1].

This research work concerns transformation from micro to macro design, as once the component micro process design is completed, the interoperability between components is the mediation for component macro design at the architecture level. The micro process of enterprise integration differentiates itself from the macro process in that the macro process is working on the configuration of components at an architectural level while the micro process deals with components both internally and in their interference in component fabrication – with how they are integrated in micro mechanisms and with how they work together in provisioning the coherence of a business goal. This includes the integration of common components and service aggregation in micro processes.

Both micro and macro processes are important to enterprise integration. Our integration strategy and framework presented in a separate research paper indicated that the transformation from business context to service component cannot be covered solely by the deconstruction process; the business semantic injection in micro process is crucial because each component is responsible for a particular business rule [2]. In a component-based implementation there is no duplication or mix-up allowed in component design; if the dependency between particular components is high and they always follow the same sequence in their referential integrity, we call this a good candidate for service aggregation. Those components with a high frequency of utilization are termed common components. Both aggregated and common components need to be handled carefully.

A component-based model builds the foundation for both business and service components, storing the intelligence of mapping and interconnectivity information in a repository in metadata form and strengthening the foundation of vertical enterprise integration. There are three areas of micro process in enterprise integration: a) service component fabrication; b) component transformation within the service layer; c) interoperability in enterprise integration and service virtualization.

Literature review

A component that conforms to a given interface means that it satisfies the contract specified by that interface and may be substituted in any context to which that interface applies [3]. Service aggregation means a composite service aggregated from at least two components that follow a certain sequence and perform a combination of coarse grained services. Service aggregation is validated when the parsing statistics show a combination of services following a unique sequence frequently forming either an aggregated service or potential service-based process; the pre-built coarse grained service provides upward service provisioning and brings in its values in eliminating the integration interface and risk caused by boundary implementation [4]. Powerful mediators dynamically or statically adapt translations between the systems, shuttling mapped data back and forth

while ensuring complete independence of the internal formats of all systems involved. At first glance, this approach solves many of the aforementioned problems; the interoperating systems do not even need to be cognizant of each other, and dynamic adaptation in translation can effectively deal with changes in the internal formats of systems. For this reason it warrants further investigation [5]. Semantic interoperability operates on a data structure (metadata profile), categorical data (classification and indexing) and factual data (authority values) level. Metadata enhancement in the context of services, after initial creation, is an important activity [6]. How to enable a seamless interoperability and search among metadata schemas presents considerable challenges [7].

Common interface is also a means to achieving common components. S. Melnik proposes a universal interface which avoids common models and languages; this is the Generic Interoperability Framework which was developed to facilitate the integration of heterogeneous information systems. Within this framework individual communication protocols, data manipulation languages and data are represented in a generic manner while preserving their ontological variety; this process indicates how a mediation infrastructure can be built using canonical wrappers, operational requirements of integration and layered architecture for component design [8]. Mediations can be explicitly configured within a solution.

D. Roman et al. in their "Web Service Modeling Ontology (WSMO)" state that semantic services need to meet the following requirements: web compliance, ontology-based, strict decoupling, centrality of mediation, ontological role separation and execution semantics [9]. These requirements were addressed in our research, and the strategy is identical to that which we proposed.

Automata theory in service component design

The design of service components follows our previous strategy by using business semantic injection to take the context into the semantic and micro processes; by this means each component has its contract-level responsibility in provisioning a subset of service. This interface helps the coordination between components and metadata to build the foundation of interoperability within the service layer and between layers. In the following paragraph we take the results from component design into the service level while applying automata theory to build up the rule-based service aggregation at the enterprise level. As we are still at a pioneer stage in applying automata theory in service provisioning and aggregation, we understand this cannot be accomplished within a short period. Further research efforts will be undertaken following the current work [10].

As the objective is to build rule-based enterprise aggregated services, we introduce the "state" concept to represent the intermediate "enterprise status" between transformation and post-transformation. The state concept in our micro process transformation is not merely dealing with a component or a service, but with the service layer and its coherence of enterprise interoperability of business and information. We extend this concept and apply event-triggered service execution to leverage the reading behavior of the automata turing machine while using service aggregation of composite services to represent the language acceptable by the machine [11]. This is shown in Figure 1 (figures are at the end of the paper).

The reason we introduced and applied automata theory in micro process component interoperability was that there is similarity in transformation of service state between automata and component interconnectivity. The event initiates service components and transfers the output from their interconnectivity, and this service state transfer uses metadata stored in repository to hold the dependency and interconnectivity information; this is same as a tuning machine in state control and output analysis. The transformation therefore requires an event to incur the service with the intelligence to either predict the next service or to verify that the called service is recognized. We use the state concept discussed above to mediate the business, service event and information in the "time" dimension. The state of the service layer and the business-service-information coherence is changed once we trigger an event and the service is successfully executed. The alphabet read in automata is leveraged to service component trigger and execution while the state change is leveraged to transformation within the service layer.

We will now review automata theory. A nondeterministic finite state automaton (NFA) is nondeterministic because it may encounter a situation in which there are multiple transformations out of a state, each using the same letter. An NFA consumes a string of input symbols; each input symbol is transformed to a new state until all input symbols have been consumed. This is similar to the scenario in which we run a loop within a service; that service recursively calls itself and takes the output as input until the completion of the loop. Each use case is a finite service; once a string of services is constructed which is acceptable to the "service machine," we consume the service string until it is finished.

NFA can be expressed by 5-tuple, (S, Σ, T, s_0, a_f), consisting of:

- a finite set of states (S)

- a finite set of input symbols (Σ)
- a transition function ($T : S \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(S)$).
- a set of states s_0 distinguished as initial (or start) states ($s_0 \in S$)
- a set of states a distinguished as accepting (or final) states ($a_i \in S$)

Where P(S) is the power set of S, ε is the empty string and Σ is the input symbol alphabet. When the last input symbol is consumed, the NFA accepts it if and only if there is some set of transitions that take it to an accepting state. Equivalently, it rejects the input symbol if no matter what choices it makes it would not end in an accepting state.

Let M be an NFA so that M = (S, Σ, T, s_0, A), and Φ be a string over the alphabet Σ. M accepts the string Φ if there exists both a representation of Φ of the form $\Phi_1, \Phi_2 \ldots \Phi n$, $\Phi i \in (\Sigma \cup \{\varepsilon\})$, and a sequence of states $r_0, r_1, \ldots, r_n$, $r_i \in S$, thus meeting the following conditions:

1. $r_0 \in s_0$
2. $r_i \in T(r_{i-1}, \Phi i)$, for i = 1, ..., n
3. $r_n \in a_i$

We take the example of Figure 2 for automata implementation in component parsing

M = (S, Σ, T, s_0, A) where $\Sigma = \{\Phi i\}$ for i = 1,2 ..., n

$S = \{ S_{ij} \}$ for i = 1 . . 6, and j = 1 . . 3 => the union of state

$\Sigma = \{ s_0, s_1, s_2, s_3, s_4, s_5 \}$ => the service component is the alphabet for reading

$r_0 = \{ S_{11}, S_{42}, S_{61} \}$ => the initial state

$r_i \in T(r_{i-1}, \Phi i)$, for i = 1, ..., n => the sequence state

$r_n = \{ S_{33}, S_{51}, S_{63} \}$ => the final state

$E = \{ \varepsilon i \}$ for i = 1, ..., n. we replace $T(r_{i-1}, \Phi i)$ by 'εi' which represents a semantic event-driving state from r_{i-1} to r_i by using service Φi mediation.

Let S^* denote the union of the concatenation of S with a frequency higher than 1, and $F(\Phi)$ denote the frequency of string Φ in the parsing process, and we assume component element c_i associated with service component s_i

When the use case is deconstructed into service and component elements in atomic form, we derive the following scenarios:

Event aggregator and service skeleton

1. At the very beginning of a use case, the first event triggers service Φ_1

$\varepsilon^1 : r_0 \times \Phi_1 \rightarrow r_1$

2. When the service event Φi (i <>1) is read and triggers the service component, we express the state transformation as follows:

$\varepsilon^i : r_{i-1} \times \Phi_i \rightarrow r_i$. and we further derive the transformation as follows:

$\varepsilon^{i+1} = (r_i) \times \Phi_{i+1} \rightarrow r_{i+1} = (r_{i-1} \times \Phi_i) \times \Phi_{i+1} \rightarrow r_{i+1}$

which is in equilibrium to $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$, and derives $\varepsilon^{sm} : r_n \times (\Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m) \rightarrow r_{m\,(n<m)}$; we call this the "semantic event aggregator."

3. As the final step of a use case, the final event trigger service Φ_n

$\varepsilon^n : r_{n-1} \times \Phi_n \rightarrow r_n$, Φ_n is a terminator, an example of which is the settlement of a daily transaction.

In any component context a formal interface contract for the invocation syntax is important. The semantics issue (the meaning of parameters and so forth) has to be solved as well (domain modeling). This is crucial to any business-to-business (B2B) and dynamic invocation scenario. Such scenarios are cornerstones of the service component vision of being flexible and agile in response to the new business needs in a world of mergers and acquisitions, business transformation, globalization and other relevant factors [12].

The string $\Phi (r_n . r_m) = \Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m$ is a path from r_n to r_m, we call the "service route," and $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$ is a "composite service" between r_{i-1} and r_{i+1} We use the symbol $\Phi_{i,\,i+1}$ to denote a composite service.

When frequency $F(\Phi_{i,\,i+1}) = F(\Phi_i \times \Phi_{i+1})$ is higher than threshold σ, we aggregate the concatenated alphabet (part of the string) $\Phi_i \times \Phi_{i+1}$ into $\Phi_{i\,,\,i+1}$, this implies that the original state r_i between r_{i-1} and r_{i+1} will no longer exist for this composite service.

When the business architect checks the results from semantic event aggregator – "ε^m" (which is an upper-layer use case scenario), service route - $\Phi_{i,\,i+1}$ (which is in the service layer) and its frequency - $F(\Phi_{i,\,i+1})$, if $F(\Phi_{i,\,i+1})$ is much higher than the threshold σ, then from the business perspective the event aggregator ε^m either consists of many composite services or is business critical. This means the aggregation service is key within the business and is crucial in daily operation from the service perspective, and that fine tuning is feasible from a multi-fine-grained boundary to a coarse-grained service. We call this the "main route" of the enterprise skeleton; in other words this is the highway between A and B. Virtual coarse-grained service eliminates the boundary and leverages many fine-grained components which can be easily configured by adding or modifying the component dynamically to achieve a broader coverage of main

route services. For this reason the main route is one of the crucial services in enterprise integration. One example is the incorporation in the telecommunications industry of an open service account, an open billing account and provisioning options for a new customer. This constitutes a main route for "new customer service" coarse-grained service, though it is aligned with fine-grained components to support customer preference, service criteria and billing options, which are branches within this main route.

In service route $\varepsilon^{non}: r_n \times (\Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m) \rightarrow r_{m,(n<m)}$ if there exist in a composite service Φ_i that r_i is omitted ($n < i < m$). All the above terminology is shown in Figure 2 regarding the formation of composite, alternatives and main route. All of these critical options are enabled by the automata model.

Conclusion

Enterprise interoperability is built on the foundation of a validation process of decision, design and provisioning stages. In this paper we focused our research on micro process design; this encompasses component-based design and interconnectivity, transformation of service aggregation through rule-based design, service provisioning in a particular dimension and service virtualization.

In transformation of service aggregation we explored the possibility of applying automata theory in service aggregation. We found that the concept of automata theory can be applied to service aggregation in building up rule-based aggregated services if we handle the service execution by using the same concept of symbolic reading. Handling the parser in use case parsing by using the same machine reading concept, the state concept can also be applied to the state of the service layer with metadata alignment to the business and information layers.

Automata theory can be applied to micro process design for component interoperability, as they have similarity in service state transformation and metadata reading. The service state uses metadata stored in repository, to hold the dependency and interconnectivity information in state control and output analysis.

The transformation therefore requires an event to incure the service with the intelligence to either predict the next service or to verify that the called service is recognized. We use the state concept discussed above to mediate the business, service event and information in the "time" dimension. The state of the service layer and the business-service-information coherence is changed once we trigger an event and the service is successfully executed. The alphabet read in automata is leveraged to service component trigger and execution while the state change is leveraged to transformation within the service layer.

A nondeterministic finite state automaton (NFA) is nondeterministic because it may encounter a situation in which there are multiple transformations out of a state, using the same letter. An NFA consumes a string of input symbols; each input symbol is transformed to a new state until all input symbols have been consumed. This is similar to the scenario in which we run a loop within a service; that service recursively calls itself and takes the output as input until the completion of the loop. Each use case is a finite service; once a string of services is constructed which is acceptable by the "service machine," we consume the service string until it is finished. The standardization of metadata technology and common services and components are all requirements in this stage.

Certain areas in this research, such as applying automata theory in forming aggregated services, are still in the pioneer stage; we will extend our research in the near future to provide more guidelines for academic workers who are interested in this area.

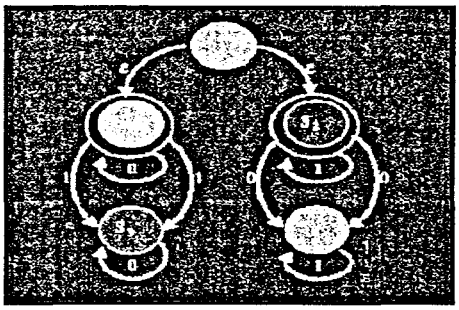


Figure 1. NFA automata theory

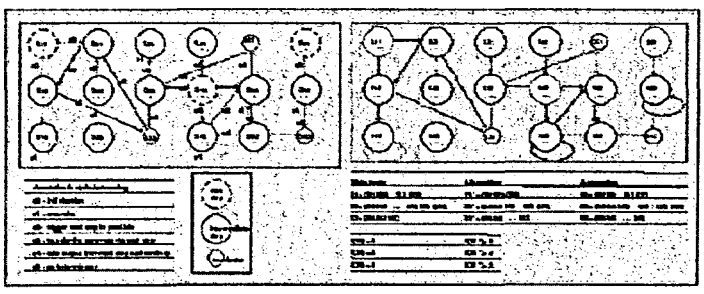


Figure 2. Service routes parsing in automata scenario

References

1. Doughty, D. (2004) "The Achilles' Heel of Service Oriented Architectures". *Business Integration Journal*, (September), 44-47.
2. Zhu, H., Madnick, S. and Siegel, M. (2002b) "Global Comparison Aggregation Services", 1st Workshop on E-Business, Barcelona, Spain.
3. Philippe Kruchten, (1998), "Modeling Component Systems with the Unified Modeling Language", 1998 International Workshop on Component-Based Software Engineering
4. S. Dwarkadas K. Gharachorloo L. Kontothanassis D. Scales M. L. Scott R. Stets, (1998) ," Comparative Evaluation of Fine- and Coarse-Grain Software Distributed Shared Memory", Cambridge Research Laboratory
5. David Wang, (2000), "Automated Semantic Correlation between Multiple Schema for Information Exchange", MIT Sloan School of Management.
6. Traugott Koch, UKOLN, (2006), "Electronic thesis and dissertation services: semantic interoperability", E-Thesis Workshop, Amsterdam 2006-01-19/20
7. L. Chan, (2005), "Metadata Interoperability: A Study of Methodology", 3rd China-US Library Conference
8. Sergey Melnik, (1999), "Generic Interoperability Framework", Distributed and Parallel Databases Journal 1999
9. D Roman et al, (2005), "Web Service Modeling Ontology (WSMO)", European Commission under the projects DIP 2005
10. R. Barcia and J. Brent, (2005), "Building SOA solutions with the Service Component Architecture", IBM Research
11. Vidal, E.; Thollard (2005), "Pattern Analysis and Machine Intelligence", IEEE Transactions on Volume 27, Issue 7, July 2005 Page(s):1013 – 1025
12. O. Zimmermann, P. Krogdahl, C. Gee, (2004), "Elements of Service-Oriented Analysis and Design", IBM Academy

M4: Service Aggregation and Coarse Grained Based Design

Publication: IC SSSM'07 (IEEE) Conference Procedeing
22 -25 March 2007, Richmond, VA, USA (http://www.southeastcon.org/2007)
ISBN: 1-4244-1029-0
Publication Contact: Professor Ou Xiaojia
Leader ICSSSM07
UESTC, Chengdu, China
ouxiaojia@uestc.edu.cn

IC SSSM'07
IEEE

From: "icsssm07" <ouxiaojia@uestc.edu.cn> Add to Address Book Add Mo
To: rwu_academic@yahoo.com
Subject: ICSSSM07 Acceptance Letter
Date: Tue, 20 Mar 2007 01:14:36 +0800

The Fourth IEEE Conference
Service Systems and Service Management
June 09-11, 2007, University of Electronic Science an
Chengdu, P. R. CHINA

University of

RE: Paper No.**A050**

TITLE: ※**Service Aggregation and Coarse Grained Based Design**§ by: **Raymo**

Dear ***Raymond Cheng-Yi Wu***,

On behalf of the ICSSSM2007 Program Committee, we are pleased to inform presentation at **The International Conference on Service Systems and Service** Chengdu, China. Congratulations! The acceptance is conditional, upon receipt of t

I would like to take this opportunity to thank you for submitting your paper to the in Chengdu.

Yours sincerely,

Yong Zeng
Conference Chair, ICSSSM2007

The Fourth IEEE Conference on
Service Systems and Service Management (ICSSSM2007)

June 09-11, 2007, University of Electronic Science and Technology of China
Chengdu, P. R. CHINA

Professor Yong Zeng

Conference Chair, ICSSSM2007
School of Management
University of Electronic Science and Technology of China
Chengdu 610054, P. R. China

2007-3-20

RE: Paper No. **A050**

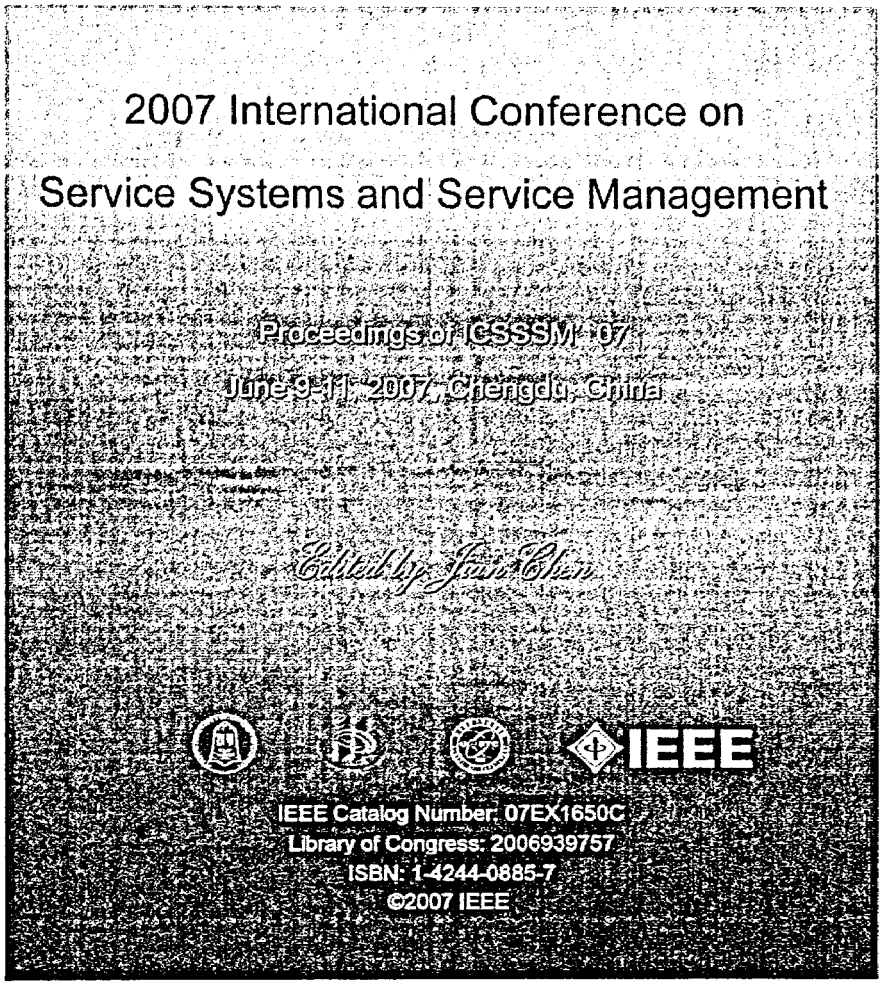
Paper Review for ICSSSM'07

Title: Service Aggregation and Coarse Grained Based Design
Authors: Raymond Cheng-Yi Wu
Paper Number: A050
Global evaluation: (Please, put "X" in the corresponding case)

	Excellent	Good	Average	Weak
Bibliography	X			
Contribution	X			
Originality		X		
Practical interest	X			
Clarity	X			
English	X			
Respect of the conference format		X		
Numerical experiments (if necessary)		X		

Dear Raymond Wu ,
On behalf of the ICSSSM2007 Program Committee, we are pleased to inform you that your paper has been published in the proceedings of ICSSSM'07.

ICSSSM07 Organization Committee
Phone: +86-28-83206652
Fax: +86-28-83206652
E-mail: ouxiaojia@uestc.edu.cn
School of Management, UESTC
Chengdu, 610054, China



PART IX Service Information Systems: Design, Implementation and Adoption

Research on the Model of GCM and IT Adoption — Chong Wang, Yuanqiang Xia, Jiaming Fang

Returns Management Information System: A Case of Paint Manufacturers — Qiaoyun Yun, Bin Liu

Service Aggregation and Coarse Grained Based Design — **Raymond Wu**

Sharing Knowledge and Creating Knowledge in Organizations: the Modeling, Implementation, Discussion and Recommendations of Weblog-based Knowledge Management — Jingjing Li

Software Metric Estimation: An Empirical Study Using An Integrated Data Analysis Approach — Da Deng, Martin Purvis, Maryam Purvis

The Design and Realization of a Medical Image Management System Based on the IDL Platform — Xinwu Li

Service Aggregation and Coarse Grained Based Design

Raymond Wu
IBM Software Group
raymond@de.ibm.com

ABSTRACT

This research deal with Analytical Model to take the result from group decision for further development, as the integration interoperability of components requires both macro process in business context and semantic metadata, and micro process within service design and physical implementation. Common service and service aggregation are conceptual term with alignment to both business components and physical objects; these can be either functional or non-functional elements. The service aggregation parsing process identifies these two services as candidates for common service; they are validated once the frequency passes the appropriate threshold and their interconnectivity design is seen as feasible.

1. INTRODUCTION

In the terminology our use of "common" indicates an encapsulated enterprise service that is seldom changed. Therefore, those fine grains at the bottom of the service layer can form a different pattern in supporting the coarse grain, and it is very often the service layer uses coarse grains to support business components directly; This implies that the fine grains serving as the foundation of the services can be reused so that one service can contribute to provisioning many coarse grains through different patterns. The formation of a pattern can use common components as a base as they are common in provisioning generic services in each business component, and the common component, once their interoperability is firm in certain patterns, they are qualified to be aggregated into coarse grain service to provision a dimensional service and to minimize the gap between business and services.

Grain size is critical in component evaluation as in non-component based environment, a minor change can cause significant impact and does not need to evaluate component grain size. In component based development, as all the services are in atomic level initially, the aggregation process construct service hierarchy, so coarse grain has opportunity to mediate business and service in eliminating the gap and shorten time to market.

Validation process and component analysis of business components includes: i) Validation process of service components in building service architecture ii) Validation process of physical components for functional and non-functional elements iii) Common component decision-making through use case statistics and parsing analysis iv) Producing service aggregation through use case statistics and parsing analysis v) Determining implementation priority and skeleton within enterprise integration. The service aggregation and parsing process identify these two services as candidates for common service; they are validated once the frequency of their appearance and co-existence passes the appropriate threshold and their interconnectivity design is seen as feasible.

The coarse grained based design requires both micro and macro process which means, when business component is split into use case or business element, all business elements are further decomposed into atomic level component and no further details can be expected. This means, we need macro architecture view when we decompose the component, and need micro interconnectivity insight when we analyze the candidacy of service aggregation.

We use the following denotation to represent the key concept that coarse grained component has less combination option than fine grained component, meanwhile the integration boundary is less than fine grained as volume is condensed and smaller than the total sum of fine grained.

$$S^k < S^k_1 \times S^k_2 \times S^k_n \quad \text{and} \quad S^k < S^k_1 + S^k_2 + S^k_n$$

From interoperability perspective, this is advantageous as degree of freedom doesn't always gain business value if a well-aligned interoperability become firm and the loosely-coupled alignment create an alternative route of aggregation service to minimize grain boundary. The process by which two separate entities coordinate and ultimately decide on the same implicit ground rules captures the essence of interoperability. That is, in order for two processes to communicate and exchange data, they must first agree to interoperate with the same language and grammar structure.

Traditionally, interoperability has been achieved through a combination of both approaches in which two systems consciously employ mediators to translate their own internal formats to a common language

1-4244-0885-7/07/$20.00 ©2007 IEEE.

and structure understood by the other; this immediately achieves interoperability. However, this practice has many drawbacks for the systems involved, including external dependencies on the mediators by the systems when the internal structures of either system changes. Powerful mediators dynamically or statically adapt translations between the systems, shuttling mapped data back and forth while ensuring complete independence of the internal formats of all systems involved. At first glance, this approach solves many of the aforementioned problems; the interoperating systems do not even need to be cognizant of each other, and dynamic adaptation in translation can effectively deal with changes in the internal formats of systems. For this reason it warrants further investigation [1]. Semantic interoperability operates on a data structure (metadata profile), categorical data (classification and indexing) and factual data (authority values) level. Metadata enhancement in the context of services, after initial creation, is an important activity. [2]

L. Chan pointed out that the rapid growth of Internet resources and digital collections and libraries has been accompanied by a proliferation of metadata schemas. Each metadata schema has been designed based on the requirements of the particular user community, intended users, type of materials, subject domain, the depth of description and other factors. Problems arise, however, when building large digital libraries or repositories with metadata records prepared according to diverse schemas. Most users do not and should not have to know or understand the underlying structure of the digital collection. In fact, confused users are actually experiencing difficulties in resource discovery and access. How to enable a seamless interoperability and search among metadata schemas presents considerable challenges [3]. A. Sheth further addresses the point that as we go from less demanding search, browsing, and personalization to more demanding integration/portal applications to even more demanding analytical/business intelligence/knowledge discovery applications, there is a greater need for deeper semantic metadata. Moreover, a processing and application logic shift from entities/concepts to relationships is also needed. Query processing requirements become increasingly demanding for analytical applications. For example, a typical analytical application involved approximately 20 complex queries (over both ontology and metadata) to display a page with analysis, but it required a sub-second response [4].

2. LITERATURE REVIEW

In the early 90's a new type of software component came into existence. These available components that could be purchased. Component Source coined a new term of components: Open Market Components. Open Market Components are reusable software are available for purchase off the shelf. A lot of these components are based on architecture such as COM, or Java and can be purchased without having to buy as well or other types of services and they are sold as genuinely plug-n-play components.

Before the design of the service component, we take direction from the business component by using a top-down approach to derive the component design while at the same time streamlining the alignment between service components and physical components by using both top-down and bottom-up approaches. Most of the bottom-up approaches are a componentization process working from existing information components, business objects, codes, system metadata or database access protocols. Once we construct the alignment between business, service and information and have the service component design and interconnectivity completed, we actively work on service layer transformation. The reason this step is crucial is that the goal of vertical enterprise integration is to create interoperability between layers from both micro and macro perspectives, with both business and information layers working as a foundation to support the formation of the service layer which provides the actual services. Once the service component has completed its design with alignment to business and information, the focus can be shifted to micro interference within the macro view.

3. COARSE GRAINED COMPONENT DESIGN

Service provisioning is based on an assessment of frequency of use of a component and its dependency within a project. Before implementation, the strategic KPI will first be reviewed based on a project's requirements followed by business component selection and validation. Once the business components have been validated, the deconstructed service components and associated component elements will be evaluated and validated, using the same validation methodology [5].

As each project needs such an assessment to determine the composition of K-B-S-C and the significance of each component regarding its frequency and dependency in provisioning validation, there are often judgments to be made. These involve both human evaluation and use case parsing to determine the components to be used and priority of deployment. that is stored in an information repository distinct from the service components. It constructs part of the component element in the physical layer. The component element consists of physical units of functional and non-functional elements; while an agent is an example of a functional element, a business object or a security component are both considered as non-functional elements. This is shown in Figure 1. According to C. Foste et al., metadata has a key role to play in allowing systems to be built and automated.

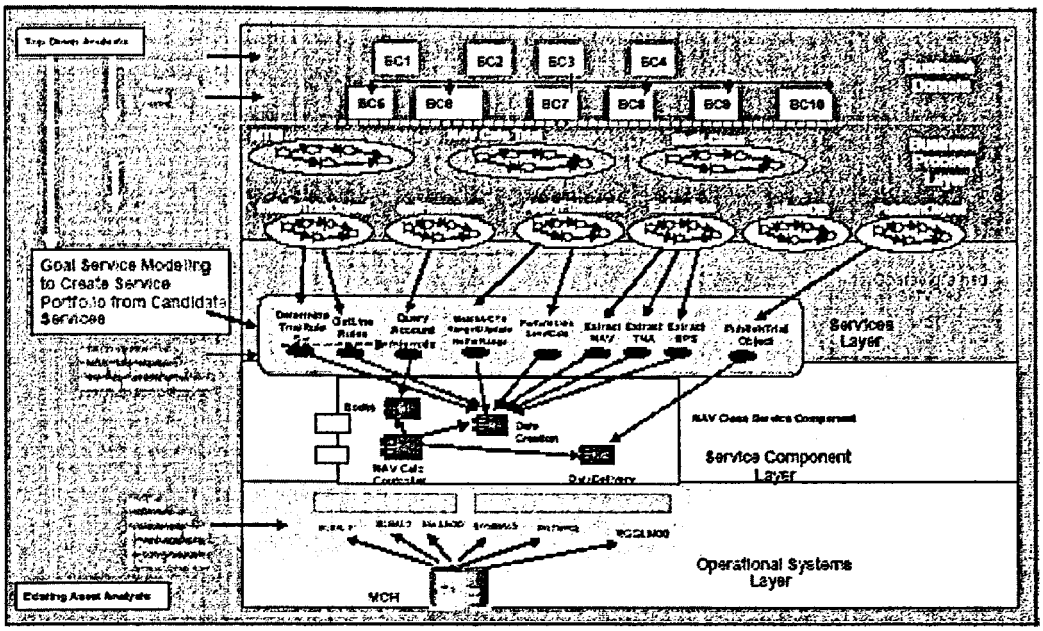


Figure 1. Coarse grained component interconnectivity

The goal of a project may be a business motivation or an IT related task; in either case an enterprise vision is required to maintain the strategic enterprise coherence of each project activity across variously compliant organizations. As mentioned, the first step is a "group decision in producing an active map" meaning that a group discussion is required for expression of opinions.

A typical group decision-making team consists of the board of the company, business manager, enterprise architect, and program manager. The information aggregation in the component layer forms the metadata

A management system composed of multiple distributed components implements a service such as content validation, transcoding, search or generation and relies on metadata information to achieve success. Metadata specification, security, transport and storage needs have been identified through the implementation of prototype components. Although the technology employed in each of the management components is often straightforward, the impact on organizational processes and interworking may be significant [6]. Service aggregation is a process for deriving the metadata within a layer, so even though the service components and component elements have been well defined in the previous 1-5 steps, fine tuning is required to derive the final step.

In Step 4 we described how the service component is aggregated. Within the component layer there is no such process; as the physical layer is at an atomic level loosely coupled with peer components and associated services, we assume all the component elements in information component form are in an optimized state and that no further aggregation is required. Therefore, before any information component is defined, a suitable aggregation in optimization is necessary; this implies that the aligned service component is always associated with an optimized component. Thus, if the component is an information component, there is no further aggregation required after S-C is configured.

In any component context a formal interface contract for the invocation syntax is important. The semantics issue (the meaning of parameters and so forth) has to be solved as well (domain modeling). This is crucial to any business-to-business (B2B) and dynamic invocation scenario. Such scenarios are cornerstones of the service component vision of being flexible and agile in response to the new business needs in a world of mergers and acquisitions, business transformation, globalization and other relevant factors. [7]

The string $\Phi(r_n, r_m) = \Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m$ is a path from r_n to r_m, we call the "service route" and $r_{i-1} \times (\Phi_i \times \Phi_{i-1}) \rightarrow r_{i-1}$ a "composite service" between r_{i-1} and r_{i+1}, we use the symbol $\Phi_{i-1, i+1}$ to denote a composite service. When frequency $F(\Phi_{i-1}) = F(\Phi_i \times \Phi_{i+1})$ is higher than threshold σ, we aggregate the concatenated alphabet (part of the string) $\Phi_i \times \Phi_{i+1}$ into $\Phi_{i, i+1}$, this implies that the original state r_i between r_{i-1} and r_{i+1} will no longer exist for this composite service. Figure 2. shows the shortcut of aggregators.

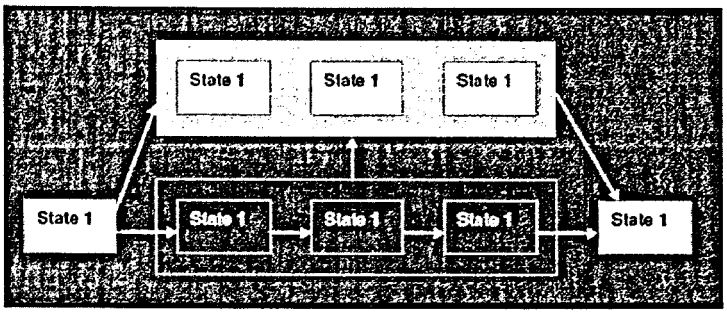


Figure 2. State aggregation

CONCLUSION

There are two type of aggregation: business driven model and component analytical model. The aggregation is validated when the analysis show a combination of services following a unique sequence frequently forming either an aggregated service or potential service-based process; the pre-built coarse grained service provides upward service provisioning and brings in its values in eliminating the integration interface and risk caused by boundary implementation. Once the aggregation services are built they can either be reused or aggregated to a coarser grain later if the aggregation is common and optimized in the parsing evaluation. These processes produce metadata interoperability for enterprise integration.

Service components and component elements by using the same method with differently weighted roles. The evaluation between service components and physical components is based on how significant their contribution to the business goal is from a logical and physical information component perspective; a higher degree of information aggregation of C within B-S-C gives a higher preference weight in that physical component. The multiplication of B and S significant factors represents the significance of S within the K-B-S-C transformation; using the same method, the multiplication of B and C significant factors represents the significance of C within the transformation. The objective of K-B-S-C transformation and significance factor evaluation is to try to build up the vertical enterprise skeleton and extend the key services to their peripheral surroundings in performing full service virtualization.

1-4244-0885-7/07/$20.00 ©2007 IEEE.

REFERENCES

1. David Wang, (2000). "Automated Semantic Correlation between Multiple Schema for Information Exchange", MIT Sloan School of Management.
2. Traugott Koch, UKOLN. (2006). "Electronic thesis and dissertation services: semantic interoperability". E-Thesis Workshop, Amsterdam 2006-01-19/20
3. L. Chan, (2005). "Metadata Interoperability: A Study of Methodology", 3rd China-US Library Conference
4. A. Sheth1, (2005), , "From Semantic Search & Integration to Analytics". Dagstuhl Seminar Proceedings 2005
5. Guangquan Zhang and Jie Lu, (2005), "The Definition of Optimal Solution and an Extended Kuhn-Tucker Approach for Fuzzy Linear Bilevel Programming". IEEE Intelligent Informatics Bulletin November 2005 Vol.6 No.2
6. C. Foster, P. Stentiford. (1999), "Metadata - The Key to Content Management Services. Proceedings of the Meta-Data", The third IEEE Meta-Data conference
7. Curtis, K., Foster, P. W., Stentiford, F. (1999), "Metadata - The Key to Content Management Services", Proceedings of the Meta-Data '99, The third IEEE Meta-Data conference, April 6-7, 1999, Bethesa, Maryland, USA

M5

M5: Decision Process in Component Based Design

Publication: ACM NZ
ISSN 1176-9998 (http://www.acm.org/chapters/acm_nz/bulletin)

Publication Contact: Taiyu Lin, Inf. Systems
Department, Massey University,
New Zealand t.lin@massey.ac.nz

Association for Computing Machinery New Zealand Bulletin

Date: Mon, 25 Jun 2007 07:07:55 +1200

From: "Taiyu Lin" <taiyu.lin@gmail.com> Add to Address Book Add Mobile
Yahoo! DomainKeys has confirmed that this message was sent by gmail.com

To: "Raymond Wu" <rwu_academic@yahoo.com>

Subject: Re: acm_nz: submissions

Dear Raymond Wu,

We are glad to notify you the accceptance of your paper titled "Decision Process in Component Based Design" by ACM_NZ Bulletin. The acceptance is based on the condition that you carry out the modification per reviewers' comments. Please find the comments at the end of this email and return your revised version to us by the end of July 2007.

Best regards,
Taiyu Lin

Decision Process in Component Based Design

Raymond Wu
IBM Software Group
rwu_academic@yahoo.com

Abstract: *The interoperability requirement in components design concerns business context, macro process, semantic metadata, micro process, and service design. This study applies validation and analysis techniques in the requirements and validate the common component and service aggregation based on the strategy and techniques it proposes. It also applies validation process and analytical model to take the result from group decision for further development; it demonstrates how to use these model and techniques to produce the common component and service aggregation which is the foundation of enterprise integration. Fuzzy interaction model has been used as supplementary to those systems that need be decided with uncertainty.*

Keyword: Enterprise, Decision support, Political model, Component design, Validation process

Introduction

The objective of decision process try to develop componentization and validation process for further development. This process is crucial as IT industry became pervasive in the 21st century due to significant growth in terms of huge volume transactions and sophisticated businesses. Enterprise architecture mainly mediates between business and IT to minimize the gap by improving governance, agility and business integrity. The component design is separated into three stages; they are briefly described as followed:

Early stage - Decision and validation process of business components, service component, physical element and Fuzzy technique in decision support.

Intermediate stage - Common component determination through use case parsing and analysis, identification of service aggregation through use case parsing and analysis.

Final stage - Component fabrication, service design and service virtualization

We start from the decision-making process of business component selection to derive the service components and physical elements. The decision-making process uses group decision-making method to take each individual opinion or preference and multiply it by a weight factor [1]. The decision makers are motivated to communicate and share opinions within the group in making business component selection. We use the same political model in all business, service and physical component validation. Political model is the human work to identify the components related to the project scope with priority for further development.

Component validation is the process to officially accept the component into project activities including the weight from evaluation. Group decision is the key process of

political model in early stage of component design. The main purpose we apply group decision in component validation was to assess and identify the impact areas starting from business components in deriving the associated service and component for further implementation. This has been the main contribution in group decision literature as this has been seldom applied in component based development, furthermore, this study took the result from component validation in common service analysis which is the small subset of ontology and subset of business components which we called active map, together with aligned service and component for further analysis to determine the common service and aggregation. The reason fuzzy techniques have been applied as a supplementary strategy of decision process was to support the validation process in uncertainties.

Literature Review

Political model in validation process has its limitation as uncertainty and dependency are all the factors affecting decision. Grabisch presented a synthesis on the application of fuzzy integral as a tool which is able to model interaction between criteria and their aggregation. Sugeno introduced the theory of fuzzy measures which related to our validation process. Our approach of validation process requires fuzzy techniques to streamline validation implementation [2]. In 1965, Zadeh, professor of UC Berkeley, extended the work on the possibility theory into a formal system of mathematical logic [3]. This new logic tool for representing and manipulating fuzzy terms was called Fuzzy Logic. A fuzzy measure g over a set X (X is the universe and E, F, ... are subsets of X) satisfies the following conditions (X is finite), from this fuzzy measure, we will be deriving the assessment of component in the decision process.

$$1)\ g(\varnothing)=0,\ \ g(X)=1$$
$$2)\ \text{If } E \subset F, \text{ then } g(E) \le g(F)$$

Enterprise skeleton is the core components derived from component evaluation, so many common components and coarse grain services have opportunity to integrate the services into portfolio foundation of enterprise common practices. Apart from fuzzy, we apply parsing technique to produce component statistics, it refer the concept of interaction measurement to our evaluation of common component and service aggregation. This construct the skeleton of enterprise interoperability between each layer and fuzzy technique is the concept in supplement our solution in the decision of component and service design [4].

Since we classify the steps in enterprise integration into three stages, the first stage covers the decision and validation and this paper focuses on the decision work in this early stage. In the following paragraphs we will determine the candidacy for common service and service aggregation by using use case parsing and modeling techniques.

As mentioned, the methodology we used in this research work included the political model and macro metadata integration; we differentiate these with conventional methods in the criteria listed in Table 1. The determination of Common Services and Composite Services is a use case driven parsing process; the purpose of the parsing iteration process

is to discover the common areas and dependency between services. The determination of common component and service aggregation, as mentioned above, is through the parsing process by using use case driven model.

Table 1. Benchmarking of integration methods in conventional and component base

Stage	Conventional implementation	Component based implementation
Decision and Analysis	Full domain	Subset of domain analysis
Architecture	Use-case driven through full domain	Component based driven and
Development	End to end dev	Component selection in a small scope
Integration	Gap exist from domain, timing and project	Alignment in both vertical and horizontal are clear
Development effort	High effort in new dev. No reuse is available	High effort in new dev. Reuse is available

The industry skeleton of common component can be constructed by the approaches this study proposed, therefore once we have this foundation built, customization or local implementation are allowed to mediate enterprise special requirements to the common services, and the philosophy in solving industry dispersed issue is, common service component and aggregation is the key of enterprise integration, once industry standardize the common service specification, we have opportunity to standardize the business component by using bottom up approach and physical implementation for component element using top down approach.

Validation process in component based environment

The initiative of enterprise integration strategy has three key tasks to fulfill: i) service validation, ii) service transformation and iii) service provisioning. Validation is mainly based on the assessment of business criteria via human decision-making with the scope of validation covering each layer of *B* (business), *S* (service) and *C* (component element). Once business components are selected for a project with alignment to *K* (strategy), the derived services from *B* will be assessed and validated. The semantic metadata within *K-B-S-C* therefore provides a means to hold their matrix alignment and this hierarchical structure constructs a vertical interoperability skeleton in which the different roles work together [5][6]. Service provisioning is based on an assessment of frequency of use of a component and its dependency within a project. Before implementation, the strategic KPI (key performance index) will first be reviewed based on a project's requirements followed by business component selection and validation.

These involve both human evaluation and use case parsing to determine the components to be used and priority of deployment. The goal of a project may be a business motivation or an IT related task; in either case an enterprise vision is required to maintain the strategic enterprise coherence of each project activity across variously compliant organizations. As mentioned, the first step is a "group decision in producing an active map" meaning that a group discussion is required for expression of opinions. A typical group decision-making

team consists of the board of the company, business manager, enterprise architect, and program manager.

There are two factors directly influencing the formation of an optimal group decision, the first being the role of the individual. Although each decision maker tries to influence other members to adopt his or her viewpoint, powerful members will sway the decision-making more strongly than will other members. Group members thus have different "weights" in a group decision-making process; this fact should be taken into account in the generation process of the satisfactory group decision. The second factor is the individual's preference for specific components. The final decision on a business component and its associated service elements needs vision from corporate strategy, business goal and enterprise architecture in order to ensure that all the different viewpoints shed light on a common focus.

Table 2. Weight granted for each role for different layer

Layer	Typical role	K-B	B-S	S-C	S-S
Strategy	Stack holder	H	L	N	-
Business	Business Mgr	H	M	N	-
Service	Architect	L	M	L	-
Component	Developer	N	L	H	-
Enterprise	Enterprise Arch.	M	H	M	-
Project	Project Mgr	L	M	M	-
K: strategy KPI, B: Business, S: Service, C: Component					
H: high, M: medium, L: low, N: neglectable					

This is true not only in the selection of all the components necessary for the project but also to create visibility of criticality, frequency and dependency of components within the project. Some projects may present a clear picture of fixed solutions in a given criteria set; however, many projects have alternatives which need to be assessed. For example, if the goal is to increase revenue by 30% in the year 2007, potential decisions on business components may vary widely according to individual perception and focus, creating diversity in opinion and randomly distributed components. Indeed, each opinion can be valuable as a political model in a component-based architecture helpful to business innovation.

The decision-making group is the same as in the previous step, consisting mainly of stakeholder, business manager, enterprise architect, project manager, business architect and other architects. Each role has its own specific weight in the various component layers, as shown in Table 2 (weight classification for role and layer).

The weight of each role is classified into the strongest R1(1.0) followed by R2(0.7), R3(0.4) and the weakest R4(0.1), while the weight of the preference rating is classified into P1(1.0), P2(0.8), P3(0.6), P4(0.4), P5(0.2) and P6(0). The rating system is designed for a variety of roles with a minimum weight of 0.1 to guarantee the inclusion of all required components from the various viewpoints of the different roles.

Following the business component selection and validation, step 2 takes the validation results from the GDPM (group decision political model) and continues the validation process in the service component layer associated with the business component.

Once a business component is selected, the associated service components can be derived. The weight classification of the role and preference is the same as above but the weight granted to each role is different. This is because when the focus is shifted from K-B to B-S, the more strongly-weighted role will be shifted from stakeholder to architect. Before input is gathered from each individual, the architect team (mainly the enterprise and business architects) will describe each service component to the group so that each member has a knowledge base from which to express and to determine the significance of a service component from an enterprise point of view.

The semantic translation from business to service and from service to technical needs a deconstruction process to work on those selected business components so that each business component can be deconstructed into business use cases for further breakdown [7]. Service components, derived from the business use case, are the mediation between business and technical elements; they integrate all business semantics, interfaces and technical configurations into a logical component [8] [9].

Each service component is initially at an atomic level, meaning that it consists merely of nouns and verb from a single subject, verb and object in business terms. Example is, a service called "open service account" in telecom industry, "open" (the verb), "service account" (the object) and a role (subject) who has enough privilege in this service, forms a service between the indicated semantics and the aligned components. In other words, service components guide technical implementation for the interface and provide metadata for physical deployment [10] [11]. The service layer has its importance in mediating the upper layer of business metadata and the lower layer of information metadata to construct the coherence of enterprise integration. From an architectural perspective, service can exist in user access, application, information and infrastructure while from a run-time architectural perspective, the user interface service at the upper layer needs a semantic web to make information accessible to human and software agents on a semantic basis [12] [13].

Fuzzy modeling techniques in component evaluation

We apply fuzzy technique as supplementary solution and review of our component decision and validation as the evaluation has some uncertainty need be addressed. In the fuzzy transformation, X is a fuzzy set, $K(F(x)) = X$, where $F(x)$ is an implied attribute of x, K is a fuzzy restriction on x, and X is the unary fuzzy relation which is assigned to K. With this interval regarded as our universe of discourse U, excellent may be interpreted as the label of a fuzzy subset of U which is characterized by membership function K, Score whose value is between 0 and 1.

Supposed we have three business components C_1, C_2 and C_3. Let ω^*_i be the rationalized weight and x_i be the individual value, according to Global Evaluation of Michel Grabisch

$$\omega^{GE}_{sum} = \Sigma (\omega^*_i x_i)$$

In global evaluation, we take each individual component into consideration without the review of intersection or the weak points. Figure 1. shows the relationship between these three components. The evaluation is shown in table 3. If we want to favor well equilibrated distribution without weak points, the above ranking is not fully satisfactory, in an Interaction Model,

Table 3. Result from Global evaluation

Component/Solution	Sales	Marketing	Order mgt.	Customer care	Global evaluation
	7	7	7	7	7.0
	7	6	9	8	7.9
	9	6	6	9	7.8
	9	9	8	6	7.5
	4	8	9	9	7.9

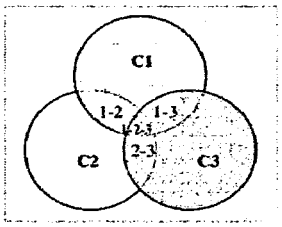


Figure 1. Relationship between components

if $n = 3$, $\omega^{IM}_{sum} = \Sigma\,(\omega^*_i\, x_i) = \omega^*_{1,2,3} \times (x_1, x_2, x_3) + \omega^*_{1,2} \times (x_1, x_2) + \omega^*_{2,3} \times (x_2, x_3) + \omega^*_{1,3} \times (x_1, x_3) +$

$\omega^*_1 \times (2x_1 - (x_1, x_2) - (x_1, x_3)) + \omega^*_2 \times (2x_2 - (x_2, x_1) - (x_2, x_3)) + \omega^*_3 \times (2x_3 - (x_3, x_1) - (x_3, x_2))$

Table 4. Result from Interaction model

Component/Solution	Sales S	Marketing M	Order mgt. O	Customer care C	Interaction model
	7	7	7	7	7.0
	7	6	9	8	7.5
	9	6	6	9	8.7
	9	9	8	6	6.9
	4	8	9	9	6.9

The reason interaction model is introduced was that too much importance is given to those similar groups which may have strong dependency between their domain interference and in a sense of redundant. Example is, let C_1 denote sales activities, C_2 denotes marketing lead and C_3 denotes OM order management, since usually, sales activities are related to marketing (and vice versa), so that the evaluation is overestimated their intersection, which can be the opportunity qualifying. The evaluation is shown in table 4.

Business component and component based implementation

Componentization starts from the decision-making process of business component selection to derive the service components and physical elements. The decision-making process uses group decision-making to take each individual opinion or preference and multiply it by the role weight factor. The decision-makers are motivated to communicate and share opinions within the group in making the business component selection in order to identify the candidacy for common service and service aggregation. Once the alignment is missing or invisible between layers, the common services and service virtualization become unfeasible, this results in a silo system leading to a serious limitation of services and weakness in enterprise integration. Two key processes were used deal with the issues. The first process relates to how the enterprise goal may be achieved by impacting business process and its service components level, which we call "business semantic." This process extends the concept of "separation of concerns" of service-oriented enterprises to impact all of the above elements. it decompose the business block into a business component; move forward to a service component through the business modeling process; the reason omponentization needed to be set first because visibility and traceability. The second process follows one level below the first one and is called "information metadata." It constitutes the repository of metadata storing information and its process of moving information between service and data; it thus supports the upper layers by repository synchronization.

When business domain is classified into several groups, each group will decompose the business block within that group while it maintain interference with other groups. Business components support the interaction model, we can either decompose business block of each group into elements and consolidate all the related business elements into business group or we can take enterprise vision to decompose business block and then do grouping. The approach develops an unique vision of enterprise business components, so business component, sales and marketing will be merged into S&M, order mgt., and customer care will be merged into CRM which is shown in Figure 2. Beneath the business components will be coarse grain service which is also the common area of sharing services which may aligned to varieties of business components.

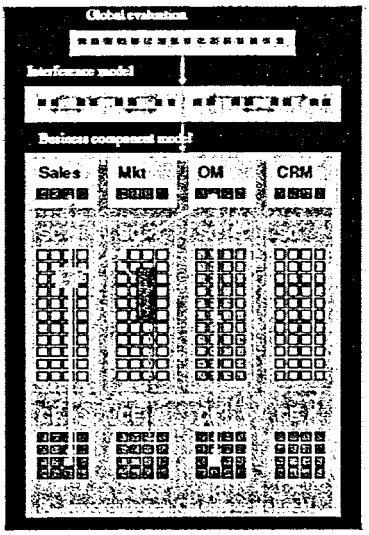


Figure 2. Business component implementation

Business component implementation further develop service component as mediator between business and technology, so the mediators can transmit semantics and maintain enterprise interoperability.

Conclusion

Political model is the foundation of validation process as the nature of business implementation requires decision input which cannot be replaced by machine or other rules, without the human effort in validation, all the analysis and implementation will be in vain. In role-based group decision-making, each role has a different weight in each layer within B-S-C and each group member has a different preference in their component selection.

Political model is used in this study to start with the business components. Political model iteration techniques have been used in a use case deconstruction to derive service components and physical elements. The group members validate both service components and component elements by using the same method with differently weighted roles. The evaluation between service components and physical components is based on how significant their contribution to the business goal; a higher degree of information aggregation of *C* within *B-S-C* gives a higher preference weight in that physical component. The multiplication of *B* and *S* significant factors represents the significance of *S* within the *K-B-S-C* transformation; using the same method, the multiplication of *B* and *C* significant factors represents the significance of *C* within the transformation. The objective of *K-B-S-C* transformation and significance factor evaluation is to try to build up the vertical enterprise skeleton and extend the key services to their peripheral surroundings in performing full service virtualization.

Fuzzy technique is applied as a supplementary of our component based development as the evaluation has uncertainty need be addressed. Each project or enterprise activity has its unique strategy, business context or KPI that first needs to be converted to business components. The alignment can reuse previous experience in a top-down K-B-S-C approach by selecting the same business components, service components and physical components to establish the business goal; however, in most project scenarios only part of the empirical K-B-S-C transformation process can be reused. The statistics of parsing results construct a matrix of common components and aggregated coarse-grained components which can be a candidate for process; this bottom-up approach attempts to create visibility of common services in either component units, interface or composite components with their priority and dependency. This provides the architects with a roadmap for building up their enterprise integration and strengthening the alignment and interoperability within the K-B-S-C transformation, and this transformation require fuzzy concept in decision support in component selection, analysis and prioritization.

Reference

1. Jie Lu, Guangquan Zhang & Fengjie Wu, (2005), Web-based Multi-Criteria Group Decision Support System with Linguistic Term Processing Function, IEEE Intelligent Informatics Bulletin June 2005 Vol.5 No.1
2. M. Grabisch, T. Murofushi, and M. Sugeno (2000), Fuzzy Measures and Integrals. Theory and Applications, Studies in Fuzziness and Soft Computing. Physica Verlag, Heidelberg, 2000
3. Lotfi A. Zadeh, King-Sun Fu, Kokichi Tanaka, Masamichi Shimura (197), Fuzzy sets and their applications to cognitive and decision, Academic Press 1974
4. J. Hunter, (2001). MetaNet - A Metadata Term Thesaurus to Enable Semantic Interoperability Between Metadata Domains, Journal of Digital Information, Volume 1 Issue 8
5. Raymond Wu, (2006), Enterprise Vertical Integration in the e-Government, Electronic Government, an International Journal (EG) ISSN (Online): 1740-7508 - ISSN (Print): 1740-7494
6. Sergey Melnik, (1999), Generic Interoperability Framework, Distributed and Parallel Databases Journal 1999
7 Sheth1 (2005) , From Semantic Search & Integration to Analytics, Dagstuhl Seminar Proceedings 2005
8. M. Staab, N. Stojanovic, R. Studer, and Y. Sure, (2001), Semantic Portal - The SEAL approach. to appear in: Creating the Semantic Web., MIT Press
9. O. Valkeapää and E. Hyvönen, (2006), Semantic Annotation with Browser-based Annotation Tool, 1st Asian Semantic Web Conference
10. Curtis, K., Foster, P. W., Stentiford, F. (1999), Metadata - The Key to Content Management Services, Proceedings of the Meta-Data '99, The third IEEE Meta-Data conference, April 6-7, 1999, Bethesa, Maryland, USA
11. L. Chan, (2005), Metadata Interoperability: A Study of Methodology, 3rd China-US Library Conference
12. Remko Scha, Rens Bod[§] and Khalil Sima'an, (1999), A Memory-Based Model of Syntactic Analysis: Data-Oriented Parsing' Special Issue on Memory-Based, Language Processing) 11, pp. 409-440.
13. S. Dwarkadas K. Gharachorloo L. Kontothanassis D. Scales M. L. Scott R. Stets, (1998) , Comparative Evaluation of Fine- and Coarse-Grain Software, Distributed Shared Memory, Cambridge Research Laboratory

M6

Componentization in SOA Development

Presentation for China Trust Corporation



Version 2.1

Facilitator: **Raymond Wu,**
Chief Technology Officer, EDS Greater China

Date: **February 12, 2006**

In this presentation, we introduced several implementation innovation and patterns in supporting componentization in SOA development. We divided the project into 3 parts,

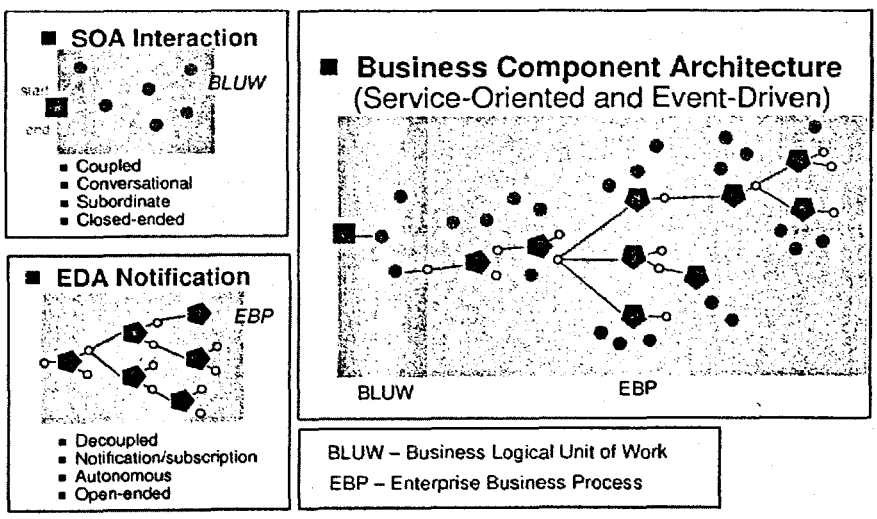
Project profile, project management and architecture.

The modern business application topology is multilayered.

Services and Web services are composed of components, built of objects.

In each layer, encapsulation is at different granularity, they are classified as followed

- *Object methods (such as C# and Java) are fine-grained and accessed in the same program context via shared memory.*
- *Components (such as EJB and .NET Remoting) may consist of hundreds of objects and are accessed via distributed computing middleware (including RPC and ORB) by other components of the same application.*
- *Service interfaces are designed to be accessible by other applications.*



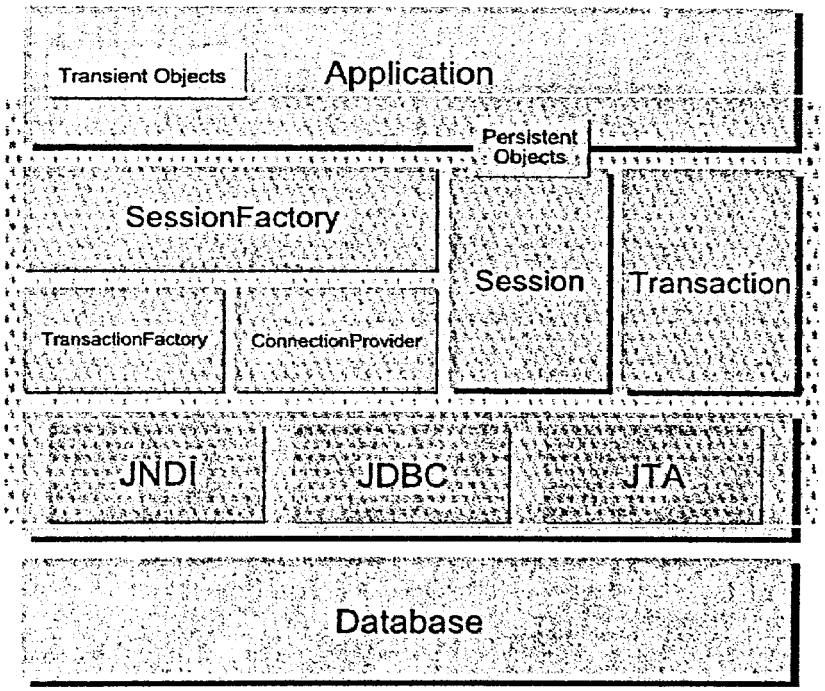
Project Profile

1. **Project Strategy**
   - **Use Case driven** - Prioritization matrix Iteration and Integration
   - **Parallelism** – Collaborative Iteration and Parallel Model Creation
   - **Component base and Object oriented** – Reuse, Loosely Couple and DCRP process
   - **Enterprise application framework** - Spring layered architecture
   - **Early Integration** – Proof of Concept from a selected business scope, early involvement for end-to-end architecture practice & common components pilot

2. **Project Management (RUP & EUP)**

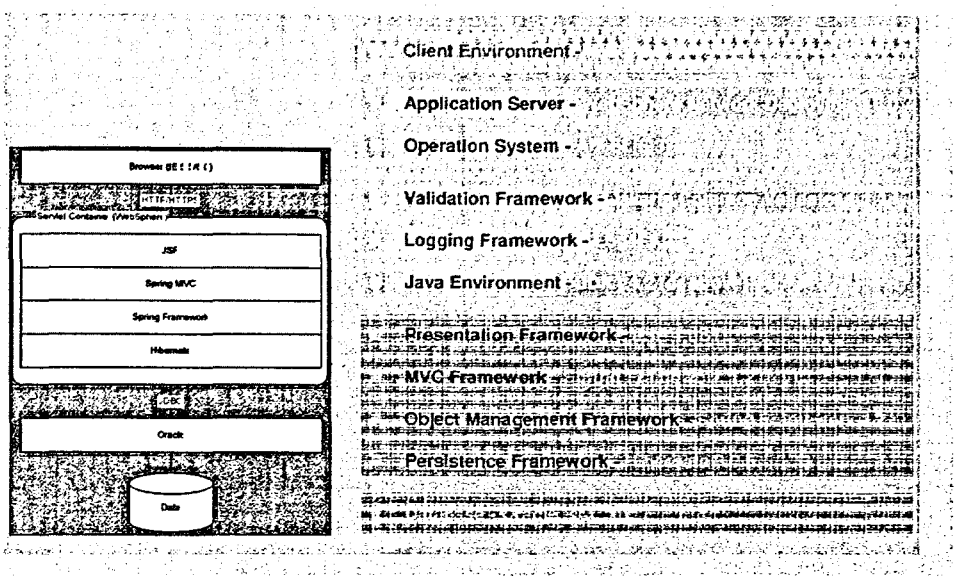
3. **Requirement Management (Use Case analysis)**

4. **Architecture**
   - **System and Data** – Infrastructure, Interface & Integration, Oracle Partition, Hibernate OR, Staging interface & ETL
   - **Application Software**– Design Pattern (MVC, Session Façade, DAO)

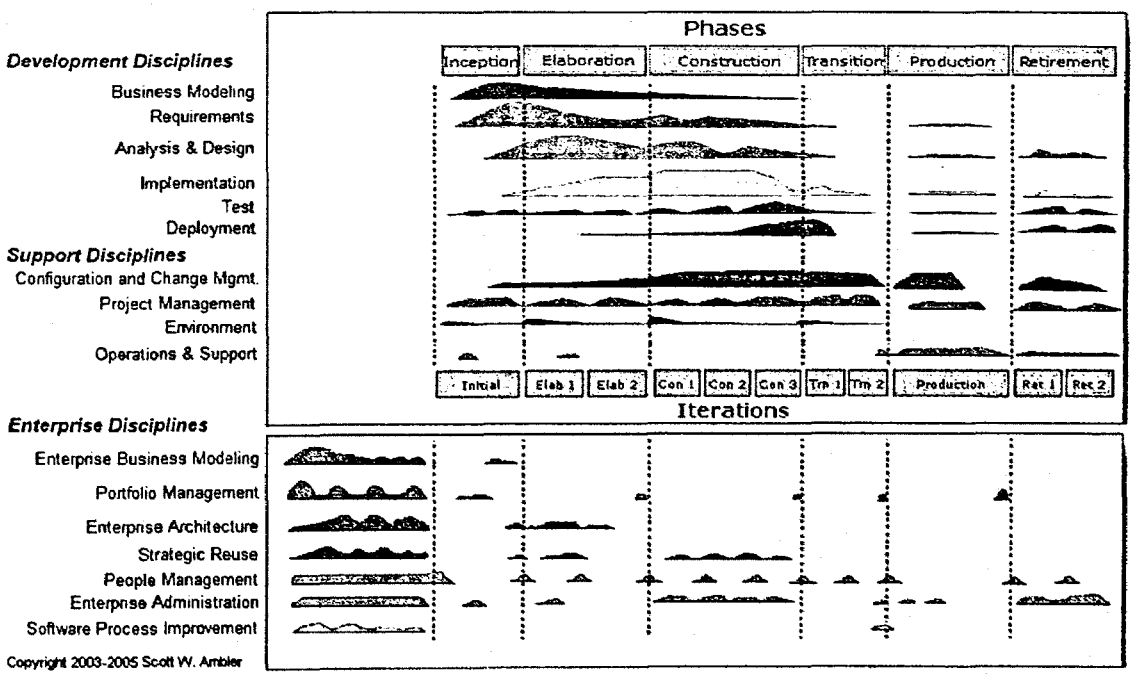


Architecture Strategy

We try to list the strategy of project management as followed



Project Management - RUP and EUP methodology



DCRP Methodlogy in Common Practices

The Reuse methodology is entitled "Decompose-Compose-Register-Provision" (DCRP), and is intended to address the labor-intensive maintenance concerns inherent in multiple instances of the CTCB Sk2

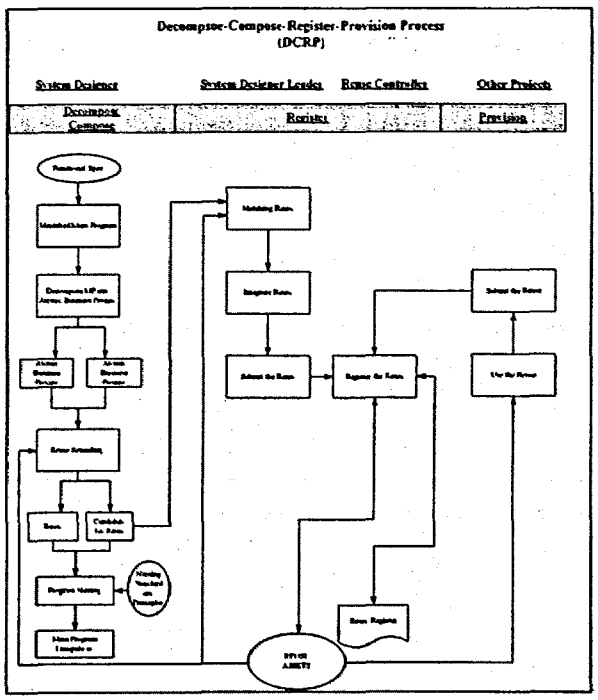
- ***Decompose*** - The functional specifications (business rules) are used to extract multiple "atomic business processes" from the Use Case.
- ***Compose*** - The "atomic business processes" are re-engineered to allow for reuse, their input and output requirements (i.e., what is passed to them and what is returned from them) are defined and documented, and their business purpose and technical requirements are captured in a searchable database. These reusable components are then crafted into a program to satisfy specific business requirements.
- ***Register*** - Reusable components are tracked within a reuse register. This register identifies every instance of employment of the reusable component (e.g., every program which contains the component).
- ***Provision*** - Reusable components are available to provision (provide) their service to any other (reusable-component-based) program which might require their functionality or business purpose.

As in Java, the method grows more powerful over time as more and more components are built and become available for reuse. DCRP is primarily intended for the construction of computer programs. The mere fact that DCRP brokers reusable components need be written only once and then reused in any and all programs .

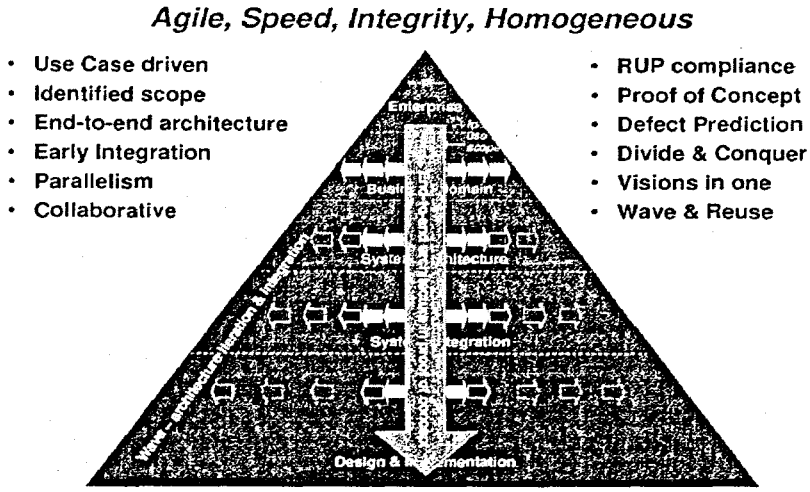
Similarly, any time a change of requirements is needed in a reusable component, it need be made only once (to the component, itself) and all of the programs that use the component can be quickly identified (through the Register) and recompiled to obtain programs containing the new requirements. The new requirements may necessitate changes to other components (e.g., changes to requisite data strings to accommodate new data fields), but these, too, would be identified through the Register and applied in a uniform (standard) manner.

Common Interface for ETL & data migration

The first step is creating a common interface for generic purpose. Working on the format and schema of this interface need a survey of requirements and existing architecture, 2nd step is we need to work on the process by using the common interface to take existing architecture to the future architecture from Data perspective.



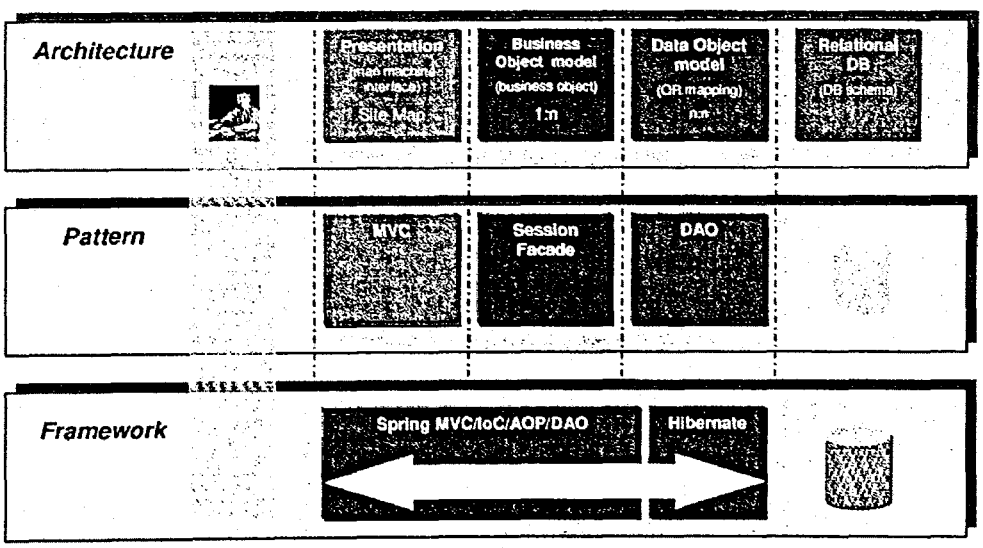
Use Case driven - Iteration, Integration and Parallelism strategy



Application Server (Web Server)	SK系統的應用伺服器 •	AIX 5L (5.3) WebSphere Network Deployment 6.0 JDK 1.4.2 (or above) Ant 1.6.2+ Spring Framework 1.2 Hibernate 3.0.5 Jakarta Common Validator 1.1.4	IBM eServer pSeries 520 - CPU*2 - Memory 8GB - 72GB HDD*2	2	100GB (SAN)
Database Server	SK系統的資料庫伺服器 •	AIX 5L (5.3) Oracle RAC 10g	IBM eServer pSeries 550 - CPU*2 - Memory 8GB - 72GB HDD*2	2	700GB (SAN)
Report Server	SK系統的報表伺服器 •	Windows 2003 Server Edition Crystal Report Server		1	

Application Architecture Strategy

Spring Framework solution can be leveraged to the following Architecture tier, Design Pattern and Framework



Spring J2EE Pattern Framework

- **Spring Framework** - Pattern framework became the priority in a high complexity of enterprise AP dev.
- **Core and IoC** - The container is the Bean Factory, it applies the *Inversion of Control* (IoC) pattern to separate an application's configuration and dependency specification from the actual application code.
- **AOP** - It incorporate declarative transaction management into applications without relying on EJB components.
- **DAO** – The abstraction layer for exception hierarchy such as opening and closing connections
- **MVC framework** - Model-View-Controller framework is a full MVC featured implementation for building Web AP

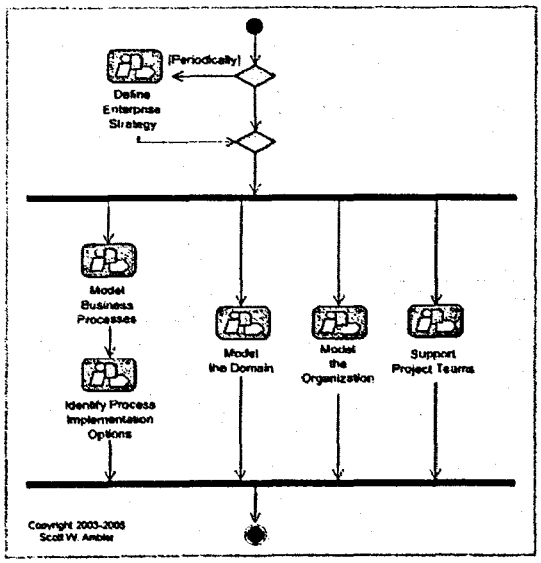
IoC, AOP and Hibernate

IoC Inversion of Control – dependency injection pattern to describe how they should be created the container creates all the objects, wires them together by setting the necessary properties, and determines when methods will be invoked.
It separate AP config. & dependency specification from actual AP code
Define Service and Component in a Configuration file

- AOP Aspect-oriented programming - allows programmers to modularize crosscutting concerns.
- The core construct of AOP is the *aspect*, which encapsulates behaviors affecting multiple classes into reusable modules.
- The advantage is that the Java class doesn't need to know about the existence of the logging service or concern itself with any related code.
- Application code written using Spring AOP is loosely coupled.
- Hibernate. A full-featured, open source OR mapping framework for the Java platform
- The Hibernate Query Language (HQL), designed as a minimal object-oriented extension to SQL through JDBC technology, is an elegant bridge between the object and relational worlds. Hibernate uses XML (**.hbm.xml*) files to map Java classes to tables and JavaBeans properties to database tables.
- Hibernate integrates elegantly with all popular JEE application servers and Web containers.

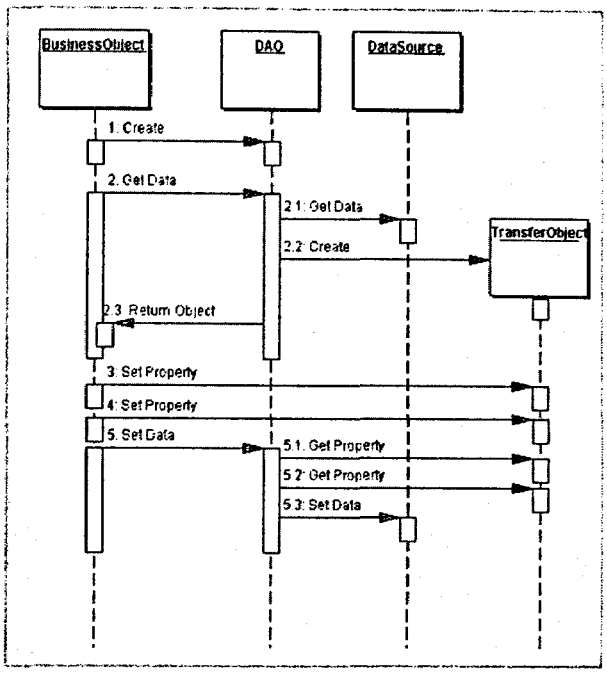
Model - Manages the behavior and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state
Use Case Prioritization matrix is based on assessment of following factors:

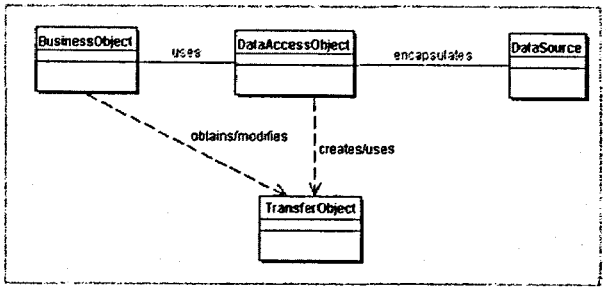
- Business Critical
- Dependency
- Effort factor
- Common factor
- Changeability factor
- Risk factor

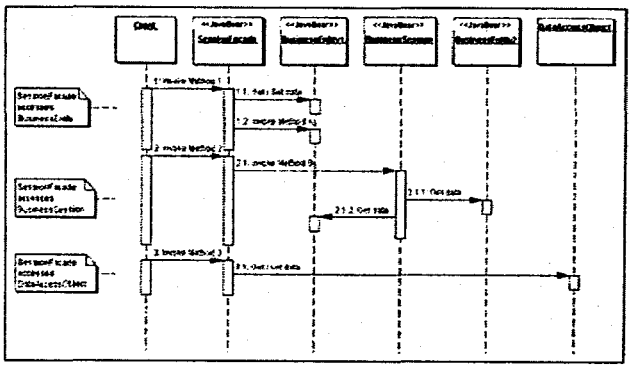


DAO

Data Access Object is designed to abstract and encapsulate all access to the data source. The DAO manages the connection with the data source to obtain and store data

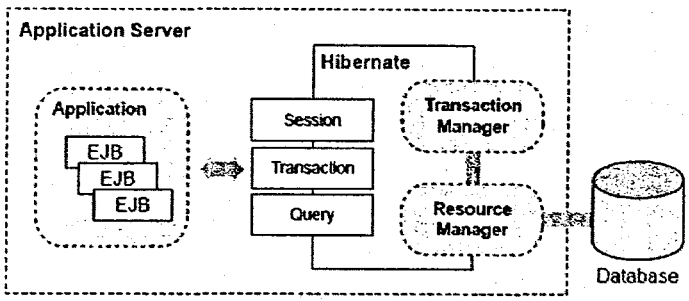


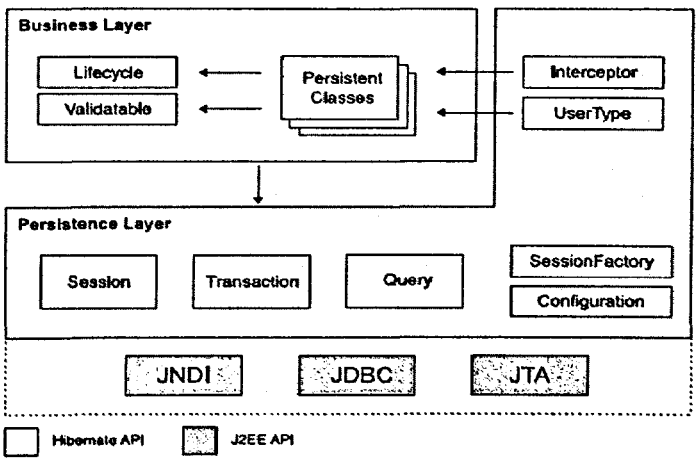




Hibernate

- Powerful, high performance
- Object & Relational persistence
- Query service
- Portable SQL extension (HQL) with an object-oriented Criteria.





Evaluation



E1 The Validation Process & Component Analysis in Ent. Integration

IJBSR

E2 The Decision and Analysis of Component Based Integration

Haamaha

E3 Validation Process and Component Analysis in Semantic Mediation

IEEE-ICIS

E4 Determination of Common Component Integration

ACM-NZ

E5 The Validation and Analysis of Component Based Metadata Integration

ACM-EATIS

E6 SOA Maturity Model Evaluation

IBM Corporation

E1

E1: The validation process and component analysis in enterprise integration

Publication: IJBSR Inderscience Journal
(http://www.inderscience.com/browse/index.php?journalCODE=ijbsr)
International Journal of Business and Systems Research
(IJBSR) ISSN (Online): 1751-2018 - ISSN (Print): 1751-200X
Publication Contact: Professor Jason C.H. Chen
Editor-in-Chief, IJBSR
chen@jepson.gonzaga.edu

Subject: Status of Paper Submission IJBSR#3339
Date: Wed, 25 Jul 2007 08:13:45 -0700
From: "Chen, Chou-Hong (Jason)" <chen@jepson.gonzaga.edu> Add to Address Book Add Mobile
To: "Raymond Wu"" <rwu_academic@yahoo.com>
CC: "Prof. Lu, Jie" <jielu@it.uts.edu.au>, "Chen, Chou-Hong (Jason)" <chen@jepson.gonzaga.edu>

-->

July 25, 2007

Dear Mr. Wu,

After a thorough double-blind review, I am very pleased to inform you that your revised manuscript entitled "**The Validation Process and Component Analysis of Enterprise Integration** " (IJBSR#3339) is scheduled for publication in a forthcoming issue of the *International Journal of Business and Systems Research (IJBSR)*, only if you fulfill the following six required actions.

With best wishes,

Jason

<><><><><><><><><><><><><><><><><><><>

Professor Jason C.H. Chen, Ph.D.

Editor-in-Chief, *International Journal of Business and Systems Research (IJBSR)*

www.inderscience.com/IJBSR

The Validation Process and Component Analysis in Enterprise Integration

Raymond Wu[*#] *and Jie Lu*[*]
[#] *IBM Software Group, Germany*
[*]*Faculty of Information Technology, University of Technology, Sydney*
P.O. Box 123, Broadway, NSW 2007, Australia

Abstract: In a rapidly changing development environment, componentization is crucial as both reuse and agility are basic requirements for maintaining a consistent responsiveness between business and IT. This study proposes a framework of metadata-based component development, aiming to achieve service virtualization through multi-disciplines and multi-steps process. It also develops service interoperability strategy which can identify and validate component easily. Based on the strategy, a roadmap is provided to analyze and synthesize components in full implementation life cycle. The proposed techniques, the multi-disciplines and dimensional services, have been applied in insurance industry and telecom sector in constructing the skeleton of service based architecture, and created service interoperability and consequently constructing the foundation of service virtualization.

Keyword: Enterprise integration, component analysis, validation process, decision making, business intelligence

Biographical notes

Raymond Wu is the IT architect currently working with IBM Software Group. His expertises include, enterprise architecture, integration, metadata, and service transformation. Previously he worked with Oracle, Citibank and Vodafone as enterprise architect.

Jie Lu is the Associate Professor of University Technology, Sydney in Australia. Dr. Lu's primary expertise is Fuzzy Techniques and she is currently leading the e-Services research Group.

1. Introduction

We are all aware that the weakness in enterprise integration has become one of the top issues in this decade as business strategy finds itself losing to its competition in service transformation. We recognize that this issue has been created by the neglect of enterprise interoperability. A confused design has thus resulted in businesses losing their enterprise governance for the accommodation of visibility in semantics, syntax and architecture alignment. In other words, enterprise has been unable to effectively leverage business, service and technology into a common vision. Consequently, any new additions or changes in the business frontier can cause significant impact which has

Copyright © 2007 Inderscience Enterprises Ltd.

led to architecture risking "re-design" instead of "re-use", and "silos" instead of "collaborative". By taking an example in "re-use" for further understanding how common service benefits enterprise interoperability, it can be presented like this, an "account balance query" may consist of a combination of "multi-account identified" functional components and "aggregated account query" non-functional components. The service aggregation parsing process identifies these two service components as candidates for coarse grain common service; they are validated once the frequency of the combination and sequence passes the appropriate threshold and their interconnectivity design is seen as feasible. Service metadata holds the mapping and routing intelligence that is stored in a service repository so that both common components and the service aggregation have an opportunity to form the main route of the common service skeleton; any business semantic translation to the process and infrastructure level becomes feasible and efficient under this skeleton.

Following introductory paragraphs envision the work this paper constitutes. The originality of knowledge contribution it proposed can be summarized as followed; the ontological analysis, the multi-discipline approaches, the synthesis of semantic interoperability and multi-dimensional interference. Before taking these concepts into further development, following breakdowns help the understanding of the foundation where this study based on:

1) An anatomy in atomic element analysis is conducted in this study. The ontology of enterprise integration consists of three dimensions; space, time and service. Space dimension denotes a scalable architecture element or technical assets. Time dimension means the activity sequence in life cycle or combinational sequence in micro component design. Service dimension stands for a business value or service patterns in virtual world. The focus of the interference between these three dimensions distinguishes this study from industry solutions in producing an originality of elemental development in service transformation.

2) By taking the three dimensions model into realization, we can easily find the interoperability coherence is missing in existing literature review instead, this study concerns the choreography of micro-macro architecture, the mediation between semantics and metadata, the synchronization between individual step and whole life cycle, and finally alignment, for how a business value aligned with service pattern. The concerns, therefore, construct a multi-disciplines approach. Among these disciplines, semantic mediation is a key in holding interoperability coherence as it is a typical example that the focus of three dimensions impact framework effectively. The details will be discussed in Section 3.

3) This study further proposes "vector" concept, instead of "object". Assumption is the fact the issues are settled is not primarily based on the component itself, instead, the study proposes to build up capability in agility in coping with chaotic world. For this reason, refining the validation and analysis process into a component synthesis is the typical example from "vector" perspective. With no doubt, component is validated and designed for re-use purpose, however re-usage concept in this study differentiates itself from conversional design as in this hierarchical structure, re-usage can be a common component in element form, or it can be an aggregated composite in service dimension. Iteration component refining helps interoperability, together with enriched-service approach, this study encompasses loosely-coupled

Copyright © 2007 Inderscience Enterprises Ltd.

and also a strong vector in semantic alignment. Consequently, it develops enterprise skeleton and potential common practices in service industry.

The above assumptions lead to the guidelines like this, enterprise interoperability is not merely a point solution; indeed the scale is so huge that we have to start with a strategic framework which is constructed by a matrix of strategy and roadmap in order to guide the whole concept into realization. All of these concerns are not merely related to the scale or complexity instead, they are more related to weak semantic interoperability, for this reason, a framework consists of validation and analysis becomes focus in this study. This paper outlines a picture of validation process and component analysis in positioning their roles in maintaining service interoperability. The result leads us to propose SCIS (Service Component Interoperability Strategy) as a strategy and META (Method, Evaluation, Technique and Application) as a roadmap to cope with the existing issues and streamline service based design. Both SCIS and META give us the opportunity to develop the concept into a quantitative analysis and industry deployment.

To understand the linkage between this study and our previous work; this study gives a brief introduction on multi-disciplines strategy (Wu, 2007); before the study moves on to the realization of these four stages, two key processes were used as supplementary architecture solution from semantics coherence perspective: The first process relates to the upper layer in determining how the enterprise goal can be achieved by impacting the business process and its service components; this is called "semantic injection." It complies with the concept of "separation of concerns" of service-oriented enterprises to impact all of the above elements. The second process follows the first one and is called "metadata collaboration." This constitutes the repository of metadata storing information and its process in coordinating messages between service and data; it thus supports the upper layers and creates visibility and traceability by repository synchronization. In supporting the approach in moving into realization, several methodologies and techniques were used, they are AHP (analytical hierarchy process) in four levels of derivation including GDM (group decision making); PMD (political model driven); RMA (rational model analysis); CCSA (common component and service aggregation) process; K-B-S-C (KPI, Business, Service and Component) transformation; parsing iteration process, semantic mediation, metadata repository and service transformation.

This study aims to answer how validation and analysis processes were applied to eliminate the risk, why it is so important to construct semantic interoperability in component based design, and how to achieve service virtualization so that the semantic interoperability can impact effectively on the industry in a full transparent environment. It is organized in seven sections. After the introduction, Section 2 is a literature review in related research areas. Section 3 introduces framework and methodology this study proposes,. Section 4 derives strategy and pattern recommendation, Section 5 aims the realization of the roadmap in achieving virtualization; and Section 6 involves qualitative modeling, for this, a study of industry scenarios is given. Finally, conclusions are discussed in Section 7.

2. Literature Review

According to Yu, an obstacle to software reuse is the large number of major modifications that frequently have to be made as a consequence of dependencies within

Copyright © 2007 Inderscience Enterprises Ltd.

the reused software components; this is the reason the study has been focused on componentization. Component-based implementation is the foundation in coping with this bottleneck (Yu, L., 2005).as it eliminates dependencies. Furthermore, the determination of common services and the composite parsing process involved in the approach are designed to discover the common areas and dependencies between services. This strategy aims at frequent modification which can be deployed in a use case driven and micro process environment. Since componentization is the first step of strategy, the advantages of componentization and the benchmarking between the component base and conventional development is listed in Table 1.

Parallel to componentization, the micro process integration is initiated by a domain analysis; in this way the analysis of the context can be interpreted into syntax with further extraction into noun and verb in order to form the common elements as domain assets (Hunter, 2001). The semantic analysis extracts noun and verb from atomic services and forms lists of this type in the semantic repository. The noun list stores customer name, id, phone, account number, and status while the verb list stores open account, check non-exist, create account, and credit check. Once the semantic analysis work is done and both verb and noun lists are stored in the semantic repository, an asset validation of common objects needs to be done before further development in a vertical (downward) direction; asset validation is a process for duplicate check, merge and final decision before all of these common objects become valid for storage in the semantic repository for reuse. The three different types of repositories (metadata, services and semantics) collaborate with each other; the strategy of "enterprise vertical integration" is based on the processes which build up these repositories. As all of the organizations share the same service means, they are all managed under the same rules; these were developed using the top-down approach. Moreover, as all of the organizations share a common protocol to communicate their semantic rules, the integration effort can be eliminated and interoperability can be enhanced through this topology once the semantic communication has been fully developed (Wang, 2000).

L. Chan has pointed out that when building large digital libraries or repositories with metadata records prepared according to diverse schemas, most users do not and should not have to know or understand the underlying structure of the digital collection. In fact, confused users are actually experiencing difficulties in resource discovery and access. For this reason enabling a seamless interoperability and search among metadata schemas presents considerable challenges (Chan, 2005). Sheth1 further addresses the point that as we go from the less demanding search, browsing, and personalization applications to more demanding integration/portal applications and then to the even more demanding knowledge discovery applications; there is a greater need for deeper semantic metadata (Sheth1, A. , 2005). This indicates that a processing and application logic shift from entities and concepts to relationships is also needed. Query processing requirements become increasingly demanding for analytical applications. For example, a typical analytical application involved approximately 20 complex queries (over both ontology and metadata) to display a page with analysis; this page, however, required a sub-second response. Hunter has argued that metadata interoperability is a fundamental requirement for access to information within networked knowledge organization systems. To achieve this goal, a common underlying data model has been developed to enable the scalable mapping of metadata descriptions across domains and media types. This model provides a set of basic building blocks for metadata modeling and recognizes the importance of "events" to unambiguously describe metadata for objects

Copyright © 2007 Inderscience Enterprises Ltd.

with a complex history. To test and evaluate the interoperability capabilities of this model, some real multimedia examples have been applied to analyze the results of mapping from this data model to a variety of different metadata domains. To overcome the limitations in semantic mapping, the metadata is capable of providing additional semantic knowledge and has been developed to enable flexible and dynamic mapping within metadata standards.

Melnik (1999) proposed a universal interface which avoids common models and languages, thus the Generic Interoperability Framework was developed to facilitate the integration of heterogeneous information systems. Within this framework individual communication protocols, data manipulation languages and data are all represented in a generic manner, thus preserving their ontological variety. This makes it possible to build a mediation infrastructure using canonical wrappers, operational requirements of integration and layered architecture for component design. Based on this common framework, this research has the opportunity to build components of both functional and non-functional elements utilizing the industry standard. DSouza delineates the standard and common component as follows: "To manage models of large scale, a coherent modeling architecture is required, with standard ways to describe shared concepts, rules, patterns, frameworks, mappings, and generators" (DSouza, D., 2001). Schmalz states that database schema and application code can be efficiently derived from various types of schema representations, particularly in the relational model, and supports comparison of a wide variety of schema and code constructs (Schmalz, M., 2003).

3. Framework and Methodology

As it is indicated in introduction, semantic mediation is an effective way to impact the interference between architecture, roadmap and service in producing a framework. In the other word, the framework this study develops, is the realization of multi-disciplines and multi-dimensions approaches, in this way the strategy can be deployed effectively in different stages of the life cycle.

3-1. Metadata-based semantics framework

Semantics approaches proposed in this research are different as conventional ones. The semantics concept we introduced consists of a three-level metadata derivation: i) the technical level; ii) the syntax level; and iii) the context level. Technical semantics focuses on how semantics can be implemented in the atomic form with an adequate metadata interface. Syntax semantics concerns the process of how the elements work together in forming the macro architecture of semantic interoperability. Context level semantics expands the semantic notation in an upward direction with linkage to the strategic context. None of these three levels can be omitted as they require semantic alignment in order to work within the overall vision. These techniques create a micro semantic process; the syntax produces the means for semantic analysis; and the context aligns implementation to the business layer. The coherence of three-level semantics metadata is missing in the existing literature review, as most of the related literature only covers one or at most two of the three levels as an approach.

Copyright © 2007 Inderscience Enterprises Ltd.

3-2. Semantic interoperability

The interoperability approach is also different from system or application level integration, as this approach is based on micro and macro process synthesis so that each atomic element has alignment with the business, service and technical layers in achieving a vertical interoperability. Meanwhile, the choreography of the atomic element form coarse-grained service and common service in achieving the macro interoperability of the enterprise. This approach is totally different from the conventional concept (Vetere, G. et al, 2005)

In the review and findings, this paper identifies issues and the challenges industry is facing. The main purpose of semantic interoperability is to construct a solid foundation for coping with those services on-demand including dynamic changes from IT, time to market, product innovation, service behavior and quality, and enterprise strategy. Metadata enhancement in the context of services, after initial creation, is an important activity in order to form a service main route (Traugott K. 2006). In this way common services can support the service grid in provisioning a transparent and highly integrated framework which has no limitations in geography, language or system. Based on this assumption, the multi-discipline approaches need to be developed into principles which until now have been missing in the literature. As distinguished from related research, the principles we proposed can be summarized as follows:

1) The philosophy of the concept is that enterprise integration should not be approached as an ad-hoc case; key issues will never be resolved unless semantic (business) and metadata (techniques) have the opportunity to be synthesized in a micro process which is the atomic component level. This vision is the foundation of the concept and is still new in this field. The study should not focus interoperability in only the system or application level; instead terms were used such as: component, service, interoperability, semantic and virtualization in a multi-dimensional integration. These dimensions consist of the service dimension, architecture in a vertical direction, project based dimension, life cycle dimension, industry framework and integration in a horizontal direction, and ending with service virtualization

2) The micro process, as mentioned above, should be further developed into a macro process in order to produce a pre-built foundation. This foundation is expected to support the architecture skeleton. The common service developed under this skeleton needs to target service virtualization which is our objective in enterprise integration. In other words, common services need to apply both fine-grained and coarse-grained service for the purpose of synthesizing them to support the macro architecture process. In taking this concept to realization, implementation needs be based on a strategic multi-dimensional roadmap which in the main consists of the validation process and component analysis. Human decisions play the same important role as in the analytical model; the interference between these two models creates semantic mediation in micro process and requires further development.

3) Service virtualization requires service interoperability in a component-based environment; this is achievable only through multi-disciplines and a multi-steps process. It was found that this vision is lacking in the literature review, as most of the research has been focused on a point solution plug-in type rather than the synthesis type ontology we proposed. Also very few examples in the existing literature encompass multi-

Copyright © 2007 Inderscience Enterprises Ltd.

dimensional views as mentioned above. Most of the related research indicates system interfacing as a point solution instead of using component synthesis in service transformation. The framework this study proposed consists of:

- ✧ Strategy: SCIS (Service Component Interoperability Strategy) works as strategy
- ✧ Roadmap: META (Method, Evaluation, Technique and Application) works as roadmap

4. Service interoperability and patterns analysis

The dimension of services also plays a key role from the enterprise or project perspective. A service may consist of both coarse-grained and fine-grained components; the best practice usually means the service is well defined in both the validation process and the component analysis as well. The maturity of service virtualization is built on top of fine-tuned dimensions of service, which is the strategy of service component interoperability.

4-1. SCIS Strategy

SCIS approach is widely used in this research which stands for the Service Component Interoperability Strategy. As mentioned, our approach in enterprise integration is a vertical direction which is also an enriched service approach in component-based implementation. This concept is crucial and is needed to be the first consideration. This implies that a thick service layer and component-based driven constitute the approach it proposed in deriving the roadmap the study proposed. We benchmark varying approaches as follow and demonstrate the reasons it have been chosen this approach.

By using a mathematical model, the study take the output from each layer denoted by a capitalized alphabetical letter:

K: KPI (key performance index), the enterprise direction and business context
B: Business, which can be expressed by use case, business or functional element
S: Service, the virtual element, mediates business component and physical component
C: Physical component, which can be functional, non-functional, or data element

Let each architecture layer be cumulative from its elements

$$B = \sum b_n \quad S = \sum s_n \quad C = \sum c_n$$

4-2. Business impact analysis

For the further understanding the service interoperability and patterns used in supporting the foundation, the study applies different patterns in demonstrating the impact they cause of different business scenarios. Enterprise integration basically takes place in the following scenarios:

- ✧ Business change: enhancement, upgrade or transformation

The business impact can be expressed like this, $B = \left(\sum b_n\right) + \left(b'_m - b_m\right)$ where b' is the new element replacing b

Copyright © 2007 Inderscience Enterprises Ltd.

✧ Adds-on: add a new business function or component
The business impact can be expressed like this, $B = \left(\sum b_{n+1}\right)$ where n+1 is the newly-added element

✧ Consolidation: merge or industry compliance
The business impact can be expressed like this, $B = \left(\sum b_p\right) + \left(\sum b'_q\right)$

A symbolic explanation is shown in Figure 3. It demonstrates the interoperability for the three patterns. We summarize industry common practices in the three patterns as described following, and explain the advantage of pattern 3 over the other two patterns:

Pattern 1 is a non-component based approach. In a conventional peer-to-peer integration, service mediation is only limited to the block level; most of the components are large block and are "glued together."

Pattern 2 is the fine-grained "glued" type. In a non-semantic component-based integration, the paper develops fine-grained interoperability between components in the construction of ad hoc type mediation; in this way the whole enterprise of different systems and applications can build their interoperability in a horizontal direction without a clear picture of the vertical direction.

Pattern 3 is the thick layer service type and is loosely-coupled in our proposal. In this full semantic mediation integration, the thick layer of service provides high agility and virtualization of the services layer in constructing a dynamic and enriched service. This pattern is a full semantic mediation provided by the thick service layer. An enriched services layer produces coarse-grained services which are aggregated from fine-grained components. Service dimensions, therefore, become an easily configured soft area for the provision of a full range of services. The notable strategies and characteristics of interoperability include: a universal schema and metadata in the information layer; semantic mediation between business, service and component; interoperability between the three repositories; and business decomposition in the upper layer.

4-3. Apply enriched service pattern to the implementation life cycle

We term Service Base Pattern Modeling (SBPM) as an analysis method of service pattern.
The conventional vertical integration can be expressed by the following union:

$\bar{O}_{output} = B \times S \times C$ means that an integration between business, process and information is a "scrambled" type without micro process visibility; the vertical integration has no chance to occur at the semantic level and there is neither micro integration extendable to the direction layer nor linkage to the service layer for non-service-oriented development. Moreover, there is no driving discipline for its output, as any small change in strategic direction can cause a macro change without micro insight.

Under the enriched service pattern, the ontology can be expressed by the following Cartesian product:

Copyright © 2007 Inderscience Enterprises Ltd.

$\tilde{O}_{output} = \sum(B_n) \cup \sum(S_n) \cup \sum(I_n)$ in conventional design

$\tilde{O}_{output} = \sum \tilde{O}_n = \sum(B_n \times S_n \times I_n)$ in decision and analysis

$\tilde{O}_{output} = \sum \tilde{O}_n = \sum(\sum b_n \times s_n \times i_n)$ in the micro design

$\tilde{O}_{output} = \sum \tilde{O}_n = \sum B^i{}_n \times \sum S^i{}_n \times \sum I^i{}_n = \sum(\sum b^i{}_n \times \sum s^i{}_n \times \sum i^i{}_n)$ in service virtualization

The conventional design should not be the option in our approach, as there is no alignment visibility between layers and components are not clear in each layer. This means that in the decision and analysis stage, the alignment interoperability B × S × I is formed with micro referential integrity and occurs in each layer with a loosely-coupled alignment between b-s-i which is the basic requirement. The concatenation between B-S-I in micro process forms a reuse foundation within enterprise and industry; the standard metadata framework within industry shares the repository for the common components (business, service and information) in order for enterprise to implement and integrate within the industry. In this way the service virtualization becomes possible. Micro process interoperability is the key in component-based integration which is supported by a synthesis process of fine-coarse grain component, as is shown in Figure 4.

5. Roadmap

META is the roadmap in this study, it projects component validation, service design and semantic mediation to the life cycle dimension. The study starts from the decision-making process of business component selection to derive the service components and physical elements. The decision-making process uses group decision-making to take each individual opinion or preference and multiply it by the role weight factor (Lu et. al, 2005). The decision-makers are motivated to communicate and share opinions within the group in making the business component selection in order to identify the candidacy for common service and service aggregation (Scha et. al, 1999). Once the alignment is missing or invisible between layers, the common services and service virtualization become unfeasible, this results in a silo system leading to a serious limitation of services and weakness in enterprise integration. In the following paragraphs we will discuss the steps for validation and an analysis of how the political model and analysis model derive component-based design and service interoperability.

Step 1 - Validation of business, service and physical components

The initiative of enterprise integration strategy has three key tasks to perform: service validation, service transformation and service provisioning. Validation is mainly based on the assessment of business criteria via human decision-making; the scope of validation covers each layer of B (business), S (service) and C (component element). Once business components are selected for a project with alignment to K, the derived services from B will be assessed and validated. The semantic metadata within K-B-S-C therefore provides a means to hold their matrix alignment; for design purposes, the referential integrity and semantic interoperability require that there be a repository to hold the information. The decision-making group is the same as in the previous step,

Copyright © 2007 Inderscience Enterprises Ltd.

consisting mainly of stakeholder, business manager, enterprise architect, project manager, business architect and other architects (within application, information and system areas). Each role has its own specific weight in the various component layers as shown in Table 2 (weight classification for role and layer). From an architectural perspective, service can exist in user access, application, information and infrastructure; from a run-time architectural perspective, the user interface service at the upper layer needs a semantic web to make information accessible to human and software agents on a semantic basis (Staab et al, 2001) (Valkeapää et al, 2006). The derivation process shown in Table 3 explains how each use case is derived for service components in the atomic form with linkage to component elements.

Step 2 - Producing common components

In the use case parsing process, first the finite Service matrix S_{ij} (i = 1, 2, 3 ... n, j = 1,2,3 ... n) is plotted in both vertical and horizontal dimensions in order to predict both the common degree and the degree of sequential dependencies. The study have designated the map produced by the plotting of a CCSA (common component and service aggregation) map in a process known as CCSA. Each service S, when plotted in the CCSA map, refers to the aggregation of service aggregation. This means that a combination of service components, if they follow a certain sequence with a certain frequency, may qualify for service aggregation; in this case, two or more than two atomic service components can be aggregated upward to a coarse-grained service in which the grain size and threshold is determined by the architecture group using either empirical benchmarking or optimization in the evaluation of advantage (i.e., coarse grain helps in minimizing interfaces) and disadvantage (loss of flexibility in the new configuration).

Step 3 - Service component aggregation

Service aggregation means a composite service aggregated from at least two components that follow a certain sequence and perform a combination of coarse-grained services. Service aggregation is validated when the parsing statistics show a combination of services following a unique sequence and frequently form either an aggregated service or potential service-based process; the pre-built coarse-grained service provides upward service provisioning and is valuable in eliminating the integration interface and risk caused by boundary implementation (Dwarkadas et al, 1998). Statistics are shown in Figure 5 as an example of CCSA parsing analysis associated with atomic service components in provisioning upward service through the alignment of S-C, S-S and B-S transformation. It introduces the structure of the transformation with focus on the S-S step. Aggregation in the service layer is important as it eliminates boundary configuration, increases the integrity of common services in an error-free configuration environment and, furthermore, can be developed into coarse-grain processes or processes adjacent to the business layer. According to Foster et al (1999), metadata has a key role to play in allowing systems to be built and automated. A management system composed of multiple distributed components implements a service such as content validation, transcoding, search or generation and relies on metadata information to achieve success. Metadata specification, security, transport and storage needs have been identified through the implementation of prototype components. Although the technology employed in each of the management components is often straightforward, the impact on organizational processes and inter-working may be

Copyright © 2007 Inderscience Enterprises Ltd.

significant. Service aggregation is a process for deriving the metadata within a layer; even though the service components and component elements have been thoroughly defined in the previous Steps 1-5, fine tuning is required to derive the final step.

6. Industry Scenarios

In the following paragraphs, this paper uses two industry scenarios as examples in presenting interoperability framework mapping between strategy and life cycle. The first scenario is in e-Government and the second scenario is in the financial industry.

6-1. e-Government scenario

Among the e-Government challenges interoperability has been identified as the most crucial. This is because the high integrity of non-system related policy and rules need to be followed across many geographical entities. Thus a high semantic requirement, robust service interfaces and the coherence of enterprise strategy are all important. Moreover, time to services for those mission critical, single source of truth of citizen based data as well as e-Government interconnectivity to other service industries are all critical; consequently these requirements reinforce the need for a fundamental change of e-Government. These called-for changes can be summarized as follows: Componentization, as mentioned above, became a trend in logical design to cope with the citizen new service concept and mission critical requirement; these either newly-added requirements or existing legacy are moving forward to a component-based architecture. There is a need to proceed through the processes of identification, specification and validation.

Global-local architecture is the other trend used in building up a topology foundation to achieve a common context, although the local authority may add its special rule leverage to global common practices. As all of the organizations share a common protocol to communicate their semantic rules, the integration effort can be eliminated and interoperability can be enhanced through this topology once the semantic communication has been fully developed.

Meta-level object definition and single source of truth are priorities in building up an efficient e-Government architecture as the integration trend in IT has been changed from peer-to-peer (n:n) to hub and spoke (1:n), and from hub and spoke to service bus (1:1). The pre-requirements for this are the consolidation of all duplicate object and data fragmentation by means of aggregation, merge or summary toward achieving a meta-level. Thus interoperability can be attained within the meta-layer to impact the individual entity effectively. Taking as an example its application in the citizen smart card scenario, it shows that "create a citizen smart card account" is a business service which consists of four composite services - "citizen medical history," "security clearance," "banking records" and "common application interface." By taking "banking records" as an example for further decomposition to atomic services, it shows that these include "loan request," "channel processing," "credit and balance check," "floor limit and fraud check," "provision service and billing profile," and "credit grant and adjust floor limit." All of these services can be analyzed for their semantic objects. The use case analysis generates a descriptive sentence of these composite services which reads as follows: "a citizen *submits a loan request* for housing" (through form or web); "administrator *processes application*"; "authority *processes check and credit review*";

Copyright © 2007 Inderscience Enterprises Ltd.

and "authority *grants and adjusts credit.*" As a top down approach, common service and common information follows the nature of a business implementation direction. From the technical perspective the idea was to eliminate the integration effort by using the "virtual organization" concept. From the organization perspective, this enhances the flexibility of service provisioning and the agility between organization and service; thus, the interoperability across organizations is increased. The realization of this top down vertical integration approach requires robust middleware to support the inter-communication between component-and-component and component-and-repository.

6-2. Financial scenario

The financial industry is one of the most complexes among the various sectors. A common rule is: the higher the occupancy of common services is within enterprise, the less suffering will take place in each IT and business transformation. As technology, business innovation and transformation create opportunity while at the same time seriously impacting enterprise if the transformation foundation is not solid, when organizations are motivated to transform their business or IT in order to survive to the next stage, any weakness in the enterprise skeleton or interoperability supported by a common component may cause high risk in each transformation. In the literature review, the study pointed out that the primary obstacle to software reuse is the large number of major modifications that frequently have to be made as a consequence of dependencies within the reused software components. Component-based implementation is the foundation in coping with this bottleneck.

A typical financial project usually consists of domain implementation, architecture and shared service. The domain implementation group is divided into several teams to handle various line of business as each team will have different areas of business domain knowledge; each team may in fact handle several business lines (i.e., use cases). It is very often the case that the individual LOB team does not have adequate concepts to determine the common components, which include common services, common interface and standard data access and data models. The study will now review and list these common issues in enterprise integration as well as propose our resolutions for them:

- Too many integration points cause implementation silos
- Data fragmentation with duplicate processes without aggregation strategy
- Common area is invisible and unable to reuse in a shard model
- Lack of priority and focus of strategy slow down business responsiveness
- Weak enterprise interoperability cause high risk in industry transformation

Under current requirements, the conventional credit card service does not maintain a robust "Fraud detection policy" at the account level, thus many transactions become invalid. The agility and responsiveness of making proper changes is still weak and many consumers suffer the serious issues that arise when their transactions need to be rolled back. For this reason, the add-on service component of Fraud Policy service was launched to replace the conventional practice and dynamically retrieve the cardholder profile and fraud policy to determine the authorization approval through inter-banking; in this way each account holder can have their security profile and preferences for the card issuer or dealers to determine individual fraud detection based on the transaction behavior and customer profile.

Copyright © 2007 Inderscience Enterprises Ltd.

7. Conclusions

In a rapidly changing environment, semantic interoperability continues to be a moving target in resolving enterprise integration. The fact is, existing literature is lack in addressing heterogeneity and rigidity issues, neither does existing architecture model accommodate disciplines to cope with either scalability or complexity problem; for this reason, a study in addressing enterprise integration issues was conducted.

The investigation led us to make following summary: 1) In this information age, the challenges of enterprise integration will only increase; unless the pending issues can be removed quickly, enterprise development will be blocked and even getting worse. To settle a complex issue such as enterprise integration is not easy; among all disciplines and solutions, method should be the first instead of merely a plug-and-play deployment as experience tells us, coping with a chaotic world can only be achieved by enabling the capability in agility and a multi-disciplines framework. 2) Semantic interoperability ensure that both humans and computer programs can be comprehended and implemented unambiguously in enterprise integration realization. Therefore the scope of enterprise integration should be enlarged and escalated to a level of interoperability in maintaining the coherence for all point solutions. 3) Unless we identify all necessary strategy, steps, patterns and interoperability, and have them represented in a generic manner preserving their ontological variety, the decisions are not effective to solve problems. To prompt broader semantic interoperability, a common framework encompassing various standards and disciplines is needed.

The research presented in this paper addresses interoperability framework, it aims to achieve service virtualization through a multi-disciplines approach. The focus of the various disciplines interference outlines a big picture in driving the interoperability concepts into realization; this paper develops META approach by projecting SCIS strategy to the validation and analysis life cycle in order to encompass synthesis process in semantic orientation.

Validation process and component analysis constitute a component synthesis process, which is the core of this framework in holding an empirical K-B-S-C alignment. This approach can reuse and tune previous experience by selecting the same business components, service components and physical components for each business context and service dimension. This study further benchmarks semantic orientation to enhance synthesis process, an enriched-service pattern which apply fine-coarse grain services approach was proposed in order to construct enterprise skeleton. The skeleton is the staging areas for common components and aggregated services which leverage common services to a particular sector; so far, the approach has been implemented in several large accounts, and helped a major insurance service provider to build enterprise skeleton by using hundred of common components produced by META approach. The originality of knowledge contribution this paper proposes requires a full involvement from key vendors and sponsorship from associated communities to move the proposed enterprise interoperability approach into industry common practices.

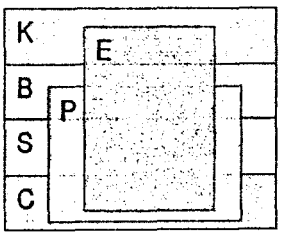
Copyright © 2007 Inderscience Enterprises Ltd.

type	Conventional development	Component base development
Analysis	Full on-demand domain	Subset of on-demand domain
Architecture	Use-case driven in full domain	Component selection and aggregation
Development	Start from scratch or adds-on	Component reuse or adds-on
Integration	Horizontal system to system	Vertical metadata to horizontal services
Future dev. effort	Med. to high as lack of traceability	Lower as "reuse" and "loosely coupled"
Service level	Low to medium - scale & constraints	Higher - service virtualization

Table 1. Benchmarking of integration methods in conventional and component base

Layer	Denotation	Role	K-B *weight (level1)*	B-S *weight (level2)*	S-C *weight (level3)*	S-S * *(level 4)*
K	Strategy	Stakeholder	R1	R3	R4	n/a
B	Business	Business Manager	R1	R2	R4	n/a
S	Service	Business architect	R3	R2	R3	n/a
C	Component	Appl. & System Architect	R4	R3	R1	n/a
E	Enterprise	Enterprise Architect	R2	R1	R2	n/a
P	Project	Project manager	R3	R2	R2	n/a

* S-C is based on the information aggregation and intensity of data access
S-S applies rationalization model in common component parsing and service aggregation
"weight" denotation – R1: 1.0/ R2: 0.7/ R3: 0.4/ R4: 0.1



Inter-Layer - Denotation	Decision method
K-B - Business component decision	Political model, human decision
B-S - Service component decision	Political model, human analysis
S-C - Information component assessment	Political model, human analysis
S-S - Common Service & Aggregation	Rational model, parsing analysis

Table 2. Weight classification for role and layer

Copyright © 2007 Inderscience Enterprises Ltd.

U1	Buyer sends payment information to merchant Merchant send and receive messages; encrypt and decrypt; store keys; and receive certificates.	E1	S1	Virtual (payment_info)	C11	Buyer_req_merchant_payment (info)
		E2	S1	Virtual (payment_security)	C12	Merchant_exec_buyer_payment (security)
		E3	S2	Online (auth_info)	C21	Merchant_exec_buyer_message
U2	Merchant software takes payment info from the cardholder and sends it to the acquirer. Acquirer receive and authenticate payment the merchant received from cardholder	E4	S1	Virtual (payment_info)	C13	Merchant_submit_acquirer_payment
		E5	S2	Online (auth_tx)	C22	Acquirer_authenticate_merchant_payment
		E6	S3	Security (merchant)	C31	Acquirer_validate_merchant
		E7	S1	Virtual (payment_merchant)	C14	Merchant_charge_acquirer_computing
U3	Acquirer sends an authorization request to the issuer over the interbank network. Issuer sends the authorization response to acquirer.	E8	S4	Cardholder (status)	C41	Acquirer_req_issuer_auth
		E9	S5	Account (3rd party)	C51	Interbank_reqs_acquirer_aut
		E10	S2	Online (auth_interbank)	C23	Issuer_respond_interbank_
		E11	S3	Security (cardholder)	C32	Interbank_check_cardholder_fraud
U4	Acquirer notifies merchant about the status of authorization; if the response is positive, merchant fulfills the order.	E12	S4	Cardholder (status)	C42	Acquirer_notify_merchant_state
		E13	S5	Account (merchant)	C52	Merchant_check_acquirer_state
		E14	S6	Recur (merchant)	C61	Merchant_fulfill_acquirer_o
U5	Merchant presents the charge to the acquirer bank	E15	S3	Security (merchant)	C33	Merchant_charge_acquirer_veri
		E16	S1	Virtual (payment_merchant)	C15	Merchant_charge_acquirer_computing
U6	Acquirer sends a settlement request to the issuer.	E17	S7	Settlement (batch_run)	C71	Acquirer_run_issuer_settler
		E18	S7	Settlement (batch_report)	C72	Acquirer_send_issuer_settlemen
U7	Issuer charges the buyer's credit card account and notifies buyer of charges.	E19	S1	Virtual (payment_issuer)	C16	Issuer_charge_by_interbank
		E20	S4	Cardholder (status)	C43	Issuer_run_report_buyer (accumulated)
U8	Buyer pays the charges to the bank. Acquirer credits merchant's account.	E21	S5	Account (buyer)	C53	Buyer_pay_issuer_interbank (credit card)
		E22	S5	Account (acquirer)	C54	Issuer_credit_acquirer_buye
		E23	S5	Account (merchant)	C55	Acquirer_pay_merchant_issuer (buyer)

Table 3. Component mapping between each layer

View Vehicle Pool Availability (VPA table)	C_{100}	DM (S) Vehicle Pool table & auto generate ext_partition_para for access option
Confirm Rental Status (CRO table)	C_{200}	DM (S/U/I) Rental Status table & auto generate ext_partition_para for access option
Vehicle Check in-out root program	C_{300}	Sync Rental Status table & Vehicle Pool table by triggering CRO UC key
Payment root program	C_{400}	DM (S/U/I) Client Mgt table & auto generate ext_partition_para for access option
Logistics root program	C_{500}	DM (S/U/I) Vehicle Mgt table
Maintain Rental Agreement RA	C_{600}	DM (S/U/I) Rental Agreement table
Batch VRC-01 (Vehicle, Rental & Client)	C_{700}	Nightly Batch process scheduled by cron job
Select Rate Range (VPA RR key)	C_{101}	DM (S) Vehicle Pool table by executing ext_partition_para RR for access option
Select Location Map (VPA LM key)	C_{102}	DM (S) Vehicle Pool table by executing ext_partition_para LM for access option
Select Time Range (VPA TR key)	C_{103}	DM (S) Vehicle Pool table by executing ext_partition_para TR for access option
Select Vehicle Type (VPA VT key)	C_{104}	DM (S) Vehicle Pool table by executing ext_partition_para VT for access option
Select Pre-paid Price Range (VPA PR key)	C_{105}	DM (S) Vehicle Pool table by executing ext_partition_para PR for access option
Select Insurance Option (VPA IO key)	C_{106}	DM (S) Vehicle Pool table by executing ext_partition_para IO for access option
Select Get Options (VPA GO key)	C_{107}	DM (S) Vehicle Pool table by executing ext_partition_para GO for access option
Report Generation for Selection criteria	C_{108}	DM (S) Vehicle Pool table & execute C_{am}
Select Customer Info (CRO CI key)	C_{201}	DM (S) Confirm Rental table by executing ext_partition_para CI for access option
Select Payment Info (CRO PI key)	C_{202}	DM (S) Confirm Rental table by executing ext_partition_para PI for access option
Update Confirmation (Update CRO UC)	C_{203}	DM (U) Confirm Rental table by executing ext_partition_para UC to activate access
Create Reservation (CRO UC='1')	C_{204}	DM (I) Confirm Rental table by executing ext_partition_para UC for access option=1
Make Reservation (CRO UC='2')	C_{205}	DM (U) Confirm Rental table by executing ext_partition_para UC for access option=2
Vehicle Check-out (RA CO='1')	C_{301}	Update Vehicle Pool Status table & auto generate RA CO script for update option=1
Sign-out Vehicle (VPS SV='1')	C_{302}	Update Vehicle Pool Status table & auto generate VPS SV script for update option=1
Check-in Vehicle (RA CO='2')	C_{303}	Update Vehicle Pool Status table & auto generate RA CO script for update option=2
Locate Rental Agreement (VPS SV='2')	C_{304}	Update Vehicle Pool Status table & auto generate VPS SV script for update option=2
Process Return Info (VPS SV='3')	C_{305}	Update Vehicle Pool Status table & auto generate VPS SV script for update option=3
Modify Rental Agreement (new)	C_{306}	Update Vehicle Pool Status table & auto generate VPS SV script for update option=9
Void Rental Agreement (VPS SV='0')	C_{307}	Update Vehicle Pool Status table & auto generate VPS SV script for update option=0
Payment Mgt CM & PM Trace	C_{501}	DM (S) Client Mgt table & Payment Mgt table
Payment Mgt CM & PM Update	C_{502}	DM (U/I) Client Mgt table & Payment Mgt table
Payment Mgt CM & PM Batch	C_{503}	C_{pm} Batch interfacing

Table 4. Component identification in "Rent a Car" services

Copyright © 2007 Inderscience Enterprises Ltd.

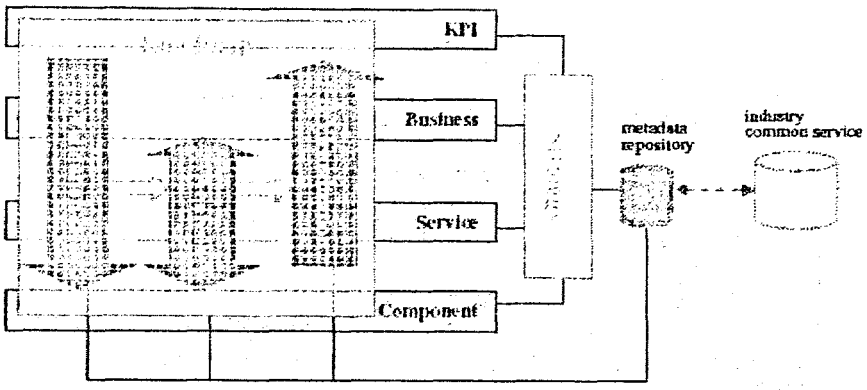


Figure 1. Framework in encompassing strategy and roadmap

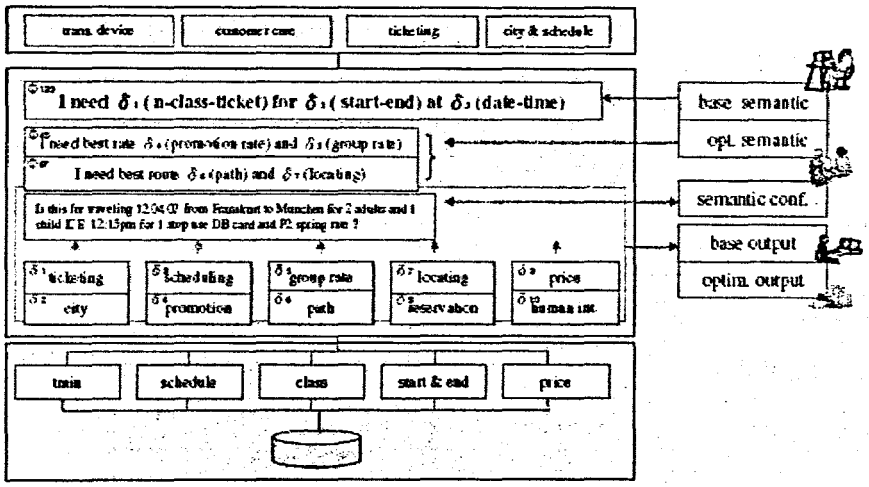


Figure 2. Semantic centric Railway reservation system

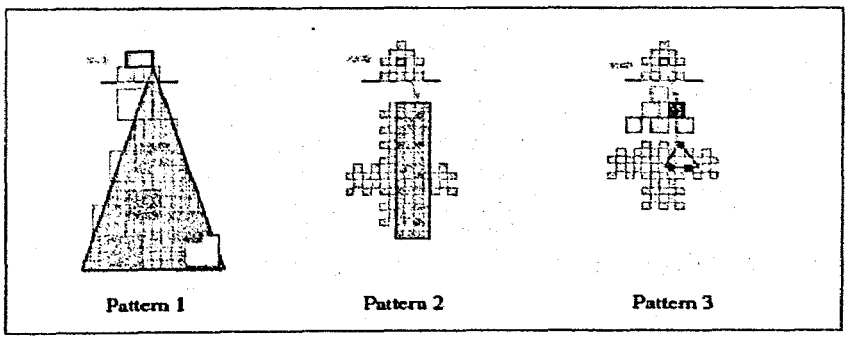


Figure 3. Benchmarking - conventional, "glued" and "loosely coupled"

Copyright © 2007 Inderscience Enterprises Ltd.

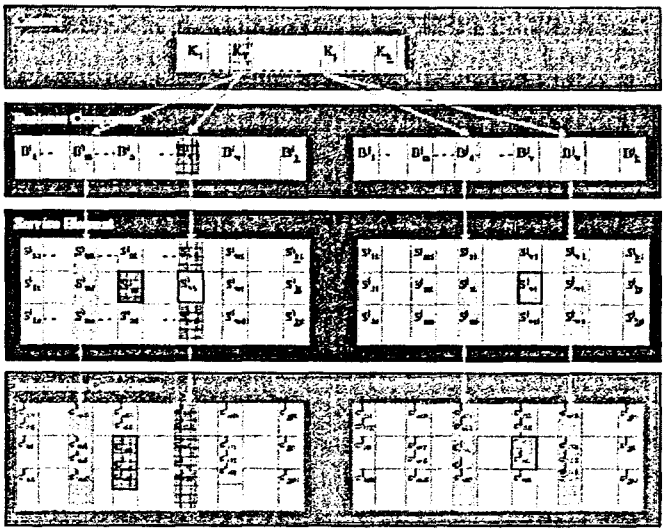


Figure 4. Referential alignment between each layer

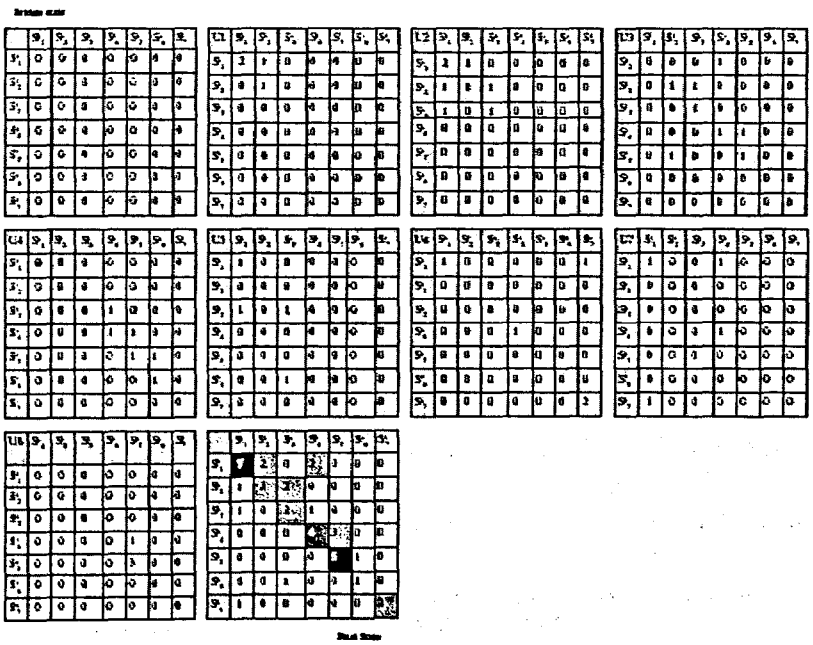


Figure 5. CCSA parsing analysis result

Copyright © 2007 Inderscience Enterprises Ltd.

References

Aciar, S., Royo, M., Josep Rosa, L., & Serarols, C., 'Increasing effectiveness in e-commerce: recommendations applying intelligent agents', International Journal of Business and Systems Research (IJBSR), vol.1 2007, pp: 26-40

Chan, L. (2005) 'Metadata Interoperability: A Study of Methodology', 3rd China-US Library Conference

DSouza, D. (2001) 'Model-Driven Architecture and Integration Opportunities and Challenges', Kinetium, pp 1-32

Dwarkadas, S., Gharachorloo, K., Kontothanassis, L., Scales, D., Scott L. and Stets, R. (1998) 'Comparative Evaluation of Fine and Coarse-Grain Software Distributed Shared Memory', Cambridge Research Laboratory

Foster C. and Stentiford, F. (1999). "Metadata - The Key to Content Management Services", Proceedings of the Meta-Data, The third IEEE Meta-Data conference, April 6-7, 1999, Bethesa, Maryland, USA

Hunter, J. (2001) 'MetaNet - A Metadata Term Thesaurus to Enable Semantic Interoperability between Metadata Domains', Journal of Digital Information, Volume 1 Issue 8, pp 234 - 253

Lu, J., Zhang, G. and Wu, F. (2005) 'Web-based Multi-Criteria Group Decision Support System with Linguistic Term Processing Function', IEEE Intelligent Informatics Bulletin 2005 Vol.5 No.1 pp 35-43

Melnik, S. (1999) 'Generic Interoperability Framework', Distributed and Parallel Databases Journal 1999

Scha, R., Bod, R. and Sima'an, K. (1999) 'A Memory-Based Model of Syntactic Analysis: Data-Oriented Parsing' Special Issue on Memory-Based, Language Processing) 11, pp. 409-440.

Schmalz, M. (2003) 'EITH – A Unifying Representation for Database Schema and Application Code in Enterprise Knowledge Extraction', 22nd International Conference on Conceptual Modeling, pp 461 - 475

Sheth1, A. (2005) 'From Semantic Search & Integration to Analytics', Dagstuhl Seminar Proceedings 2005, pp 1-10

Staab, M., Stojanovic, N., Studer, R. and Sure, Y. (2001) 'Semantic Portal - The SEAL approach. to appear in: Creating the Semantic Web', MIT Press, pp 317 – 362

Traugott K. (2006) 'Electronic thesis and dissertation services: semantic interoperability', E-Thesis Workshop, 2006-01-19/20, Amsterdam, pp 1-17

Valkeapää, O. and Hyvönen, E. (2006) 'Semantic Annotation with Browser-based Annotation Tool', 1st Asian Semantic Web Conference

Vetere, G. and Lenzerini, M. (2005) 'Models for semantic interoperability in service-oriented architectures', IBM System Journal Volume 44, Number 4, 2005, pp 887 - 903

Wang, D. (2000) 'Automated Semantic Correlation between Multiple Schema for Information Exchange', MIT Sloan School of Management

Wu, R. (2007) 'Semantic Metadata in Enterprise Integration', IEEE Southeast 2007 conference, pp 233-236

Yu, L. (2005) 'Categorization of Common Coupling in Kernel-Based Software', 43rd ACM Southeast Conference, 2005, pp 694 - 706

E2: The Decision and Analysis of Component Based Integration

Publication: Haamaha July 2007

International Conference on Human Aspect in Advanced Manufacturing and Hybrid Automation

9th-12th July, 2007, Poznan, Poland (http://www.ergonet.net/haamaha2007)

Publication Contact: Prof. Stefan Trzcieli

Chairman, Haamaha 2007

Vice Dean of Computing and Mgt

Poznan University of Technology,

Stefan.Trzcielinski@put.poznan.pl



From: "HAAMAHA 2007" <haamaha@put.poznan.pl>

To: rwu_academic@yahoo.com

Subject: Re: HA5K065

Date: Wed, 7 Mar 2007 19:16:34 +0100

Confirmation and invitation for participation at International Conference on Human Aspect in Advanced Manufacturing and Hybrid Automation " Managing Enterprise of the Future " July 9 th – 12 th , 2007. Poznan University of Technology ,

Dear Mr RAYMOND WU ,

RE: HA5K065 The Decision and Analysis of Component Based Integration

Congratulations! I am pleased to inform you that your paper(s) has been accepted for presentation at HAAMAHA ' 07. If you have co-authors whose e-mail addresses were not provided earlier, Poznan , Poland.

Sincerely Yours,

Stefan Trzcielinski

Stefan Trzcieliński, Ph.D., D.Sc. Associate Professor Vice Dean of Faculty of Computing and Management

Poznan University of Technology, Institute of Management Engineering

ltrieval system
emission from

Otganizations in Changing Environment. Current Problems,Concepts and Methods of Management

Proceedings of the E ewenth International Conference on Human Aspects of Advanced Manufacturing: Agility, and Hybrid Automation
4th International Conference ERGON-AXIA

HAAMAHA 2007

Managing Enterprise of the Future

July 9-12, 2007 Poznań, Poland

Edited by:

Wiesław M. Grudzewski
Irena Hejduk
Institute of Organization and Management in Industry

Stefan Trzcieliński Poznan
University of Technology

o persons or
lse or
operatch.

2007

IEA Press International Ergonomics
Association 610 Walnut Street 575
WARF Madison, WI 53726 USA

Modern Enterprise

1 A framework of the work active for future worker [illegible] Kim, Myung-Sub, Kim Myung-Soo, Lee Jeong-Myun, Kim Sung-Ha
2 The impact of IT value on firm performance
[illegible] companies listed in the [illegible]
[illegible]
3 Influence of participation oriented IT development on models of production organization
[illegible]
4 Micro process design in component based architecture
Wu Raymond
5 Enterprise Architecture evaluation components
[illegible]
6 The decision and analysis of component based integration
Wu Raymond
7 The [illegible]
8 Enterprise architecture for process [illegible] communication components [illegible]
[illegible]
9 Design and implementation of [illegible]
[illegible] organization's process in MEXICO
[illegible]
10 Enterprise component based interoperability [illegible]
Wu Raymond
11 Intelligent agents to support e business applications
[illegible]
12 Complex numerical algorithms for people identification
[illegible]
13 The impact of [illegible]
[illegible]
14 The process of [illegible] and [illegible]

The Decision and Analysis of Component Based Integration

Raymond Wu
IBM Software Group
+49 7031 16 1755 / raymond@de.ibm.com

Decision input from both human factors and computing analysis have equal importance in component-based implementation, with each component in business, service or physical layers forming a vertical alignment for the enterprise integration foundation. Therefore, component decision and prioritization using political and analytical models are applicable in creating the skeleton of enterprise integration. This skeleton exploits a loosely coupled approach in creating integration interoperability and constructing the foundation for service virtualization.

Keywords

Integration, Component, Enterprise, Interoperability, Semantic, Mediation

Introduction

We continue the previous strategic work [1], with further investigation of interoperability within enterprise integration. Vertical enterprise integration is a metadata driven approach in the component-based environment. There are five steps in its implementation and realization:

1. Validation of business components through political group decision-making
2. Validation of service components in building service architecture
3. Validation of physical components for functional and non-functional elements
4. Common component decision through use case statistics and parsing analysis
5. Producing service aggregation through use case statistics and parsing analysis

We start from the decision-making process of business component selection to derive the service components and physical elements. The decision-making process uses group decision-making to take each individual opinion or preference and multiply it according to the weight specified by its role. According to J. Lu, organizational decisions are often made in groups where group members may be distributed geographically in different locations in what we call "political model driven" decision-making; this occurs when group members are motivated by their own needs and perceptions [2]. The decision makers are motivated to communicate and share opinions within the group in making business component selection.

We use the same political model in all business, service and physical component validation, based expertise.

In the following paragraphs we will let 'K' denote the strategy from the stakeholder, 'B' the business component, 'S' the service component and 'C' the physical element. The

active map shown in Figure 1 demonstrates the business component selection, with the selection being based on the project's strategy K, the executives' concern; once the project defines its K, the business component needs a group decision on B to achieve K. This step will be repeated for S and C until all the relevant persons finish their assessment and we have the final result. This is followed by the political model decision, in which the enterprise architect takes the active map and validated components and continues the service component transformation we term "rational model driven" and is grounded in objectives, alternatives, consequences and optimality. In this step we use a parsing technique to analyze the use case and break it down into service components at an_atomic level so that each of the derived components consists of one and only one subject noun that to determine the candidacy for common service and service aggregation [3].

Both political and rational models are constructed in a component-based environment; this is because a conventional development environment provides no opportunity of group decision-making or rational analysis, as with all the contributing factors in a scrambled code, any changes or add-ons can have a huge impact. Even the scope of the impact area and components can be difficult to determine, thus the impact is high while the interoperability is low. In conventional development, weak visibility between business, service and information makes the linkage and enterprise interoperability impossible. Once the alignment is missing or invisible between layers, the common services and service virtualization become impossible as these require a foundation in both macro interoperability and micro mechanism. This results in a silo system with serious limitation of services and weakness in enterprise integration [4].

Service aggregation means a composite service aggregated from at least two components that follow a certain sequence and perform a combination of coarse grained services. Service aggregation is validated when the parsing statistics show a combination of services following a unique sequence frequently forming either an aggregated service or potential service-based process; the pre-built coarse grained service provides upward service provisioning and brings in its values in eliminating the integration interface and risk caused by boundary implementation [5].

Common service means a service component such as has often been used in the use case driven process. In the customer-based service industry, an "account status query" can be easily identified in many requests or processes. Once we can align this common service to both physical and business components for their unique alignment, we call it a common component. The concept of common component can encapsulate both conceptual and physical elements in provisioning a service. A common component is not limited to a functional element; any information or system driven element such as business object, portal UI element, data access protocol or component interface would qualify as a common component candidate.

Common service is a conceptual term with alignment to both business components and physical objects; these can be either functional or non-functional elements. An "account balance query" may consist of a combination of "multi-account identified" functional components and "database query" non-functional components. The service aggregation parsing process identifies these two services as candidates for common service; they are validated once the frequency passes the appropriate threshold and their interconnectivity design is seen as feasible.

Service metadata holds mapping and routing intelligence that is stored in a service repository so that common components and service aggregation have an opportunity to form the main route of the common service skeleton; any business semantic translation to the process and infrastructure level becomes feasible and efficient under this main

route skeleton [6].

The process by which two separate entities coordinate and ultimately decide on the same implicit ground rules captures the essence of interoperability. That is, in order for two processes to communicate and exchange data, they must first agree to interoperate with the same language and grammar structure. Traditionally, interoperability has been achieved through a combination of both approaches in which two systems consciously employ mediators to translate their own internal formats to a common language and structure understood by the other; this immediately achieves interoperability. However, this practice has many drawbacks for the systems involved, including external dependencies on the mediators by the systems when the internal structures of either system changes. Powerful mediators dynamically or statically adapt translations between the systems, shuttling mapped data back and forth while ensuring complete independence of the internal formats of all systems involved. At first glance, this approach solves many of the aforementioned problems; the interoperating systems do not even need to be cognizant of each other, and dynamic adaptation in translation can effectively deal with changes in the internal formats of systems. For this reason it warrants further investigation [7]. Semantic interoperability operates on a data structure (metadata profile), categorical data (classification and indexing) and factual data (authority values) level. Metadata enhancement in the context of services, after initial creation, is an important activity. [8]

In our recent research we have learned that the responsiveness within enterprise architecture has become critical; misalignment between business and IT as well as fragmentation within systems and database instances cause development bottleneck, slow down software development life cycle and time to market; this results in high impact on the coherence of enterprise integration and ROI (return on investment) which is the major concern of stakeholders.

As mentioned above, the methodology we used in this research work included the component model, political model, rational model and vertical metadata integration; we differentiate these with conventional methods in the criteria listed in Table 1.

In the following paragraph, we demonstrate how to use the six identified steps to derive the coherence of their interference in building up enterprise integration. Before we proceed to the key area of six steps, we use architecture terminology "K-B-S-C transformation" to represent interoperability between architecture layers. We began with enterprise strategy (K), the chief area of concern for stakeholders and business managers, to derive the components in both business (B) and services (S) layers. As each layer of K (context), B (conceptual) and S (logical) are all at the conceptual or logical level, the bottom layer which is the physical component (C) forms a loosely-coupled alignment with S in provisioning bi-directional interoperability. The micro mechanism between these two layers forms the service unit; aggregate individual service provides service in the business layer to finally achieve the strategic goal.

The denotation is summarized as follows:

K: KPI - strategic Key Performance Identifier and all concerns of the stakeholder

B: Business Component - Unique Business Block - each block may perform individual and independent business

S: Service element - Logical service component in atomic level

C: Component element - Physical component associated with service component - Business Object is a typical example

We will now extend our focus to each step and demonstrate the validation and analysis processes through these steps.

Step 1, 2 & 3

The initiative of enterprise integration strategy has three key tasks to fulfill: service validation, service transformation and service provisioning. Steps 1, 2 and 3 relate to service validation, steps 4 and 5 to service transformation and step 6 to service provisioning. Validation is mainly based on the assessment of business criteria via human decision-making with the scope of validation covering each layer of B (business), S (service) and C (component element). Once business components are selected for a project with alignment to K, the derived services from B will be assessed and validated. The semantic metadata within K-B-S-C therefore provides a means to hold their matrix alignment; for design purposes, the referential integrity and semantic interoperability require that there be a repository to hold the information. The K-B-S-C metadata integration forms a skeleton in enterprise integration with the choreographic capability of retrieving each layer's metadata for the details of each component stored in the metadata repository of B, S and C. This macro to micro hierarchical structure constructs a vertical interoperability skeleton in which the different roles work together; the architect takes care of the main route of the skeleton in metadata integration and the developer takes over the skeleton's work and extends the micro design in the transformation of components and interconnectivity of service.

As each project needs such an assessment to determine the composition of K-B-S-C and the significance of each component regarding its frequency and dependency in provisioning validation, there are often judgments to be made. These involve both human evaluation and use case parsing to determine the components to be used and priority of deployment. The goal of a project may be a business motivation or an IT related task; in either case an enterprise vision is required to maintain the strategic enterprise coherence of each project activity across variously compliant organizations. As mentioned, the first step is a "group decision in producing an active map" meaning that a group discussion is required for expression of opinions. A typical group decision-making team consists of the board of the company, business manager, enterprise architect, and program manager.

There are two factors directly influencing the formation of an optimal group decision, the first being the role of the individual. Although each decision maker tries to influence other members to adopt his or her viewpoint, powerful members will sway the decision-making more strongly than will other members. Group members thus have different "weights" in a group decision-making process; this fact should be taken into account in the generation process of the satisfactory group decision. The second factor is the individual's preference for specific components. The final decision on a business component and its associated service elements needs vision from corporate strategy, business goal and enterprise architecture in order to ensure that all the different viewpoints shed light on a common focus. This is true not only in the selection of all the components necessary for the project but also to create visibility of criticality, frequency and dependency of components within the project. Some projects may present a clear picture of fixed solutions in a given criteria set; however, many projects have alternatives which need to be assessed. For example, if the goal is to increase revenue by 30% in the year 2007, potential decisions on business components may vary widely according to individual perception and focus, creating diversity in opinion and randomly distributed components. Indeed, each opinion can be valuable as a political model in a component-based architecture helpful to business innovation.

Following the business component selection and validation, step 2 takes the validation results from the GDPM (group decision political model) and continues the validation process in the service component layer associated with the business component.

The decision-making group is the same as in the previous step, consisting mainly of stakeholder, business manager, enterprise architect, project manager, business architect and other architects (within application, information and system areas). Each role has its own specific weight in the various component layers, as shown in Table 2 (weight classification for role and layer). The weight of each role is classified into the strongest R1(1.0) followed by R2(0.7), R3(0.4) and the weakest R4(0.1), while the weight of the preference rating is classified into P1(1.0), P2(0.8), P3(0.6), P4(0.4), P5(0.2) and P6(0). The rating system is designed for a variety of roles with a minimum weight of 0.1 to guarantee the inclusion of all required components from the various viewpoints of the different roles.

Once a business component is selected, the associated service components can be derived. The weight classification of the role and preference is the same as above but the weight granted to each role is different. This is because when the focus is shifted from K-B to B-S, the more strongly-weighted role will be shifted from stakeholder to architect. Before input is gathered from each individual, the architect team (mainly the enterprise and business architects) will describe each service component to the group so that each member has a knowledge base from which to express and to determine the significance of a service component from an enterprise point of view.

The semantic translation from business to service and from service to technical needs a deconstruction process to work on those selected business components so that each business component can be deconstructed into business use cases for further breakdown. Service components, derived from the business use case, are the mediation between business and technical elements; they integrate all business semantics, interfaces and technical configurations into a logical component. Each service component is initially at an atomic level, meaning that it consists merely of nouns and verb from a single subject, verb and object in business terms. In other words, service components guide technical implementation for the interface and provide metadata for physical deployment. The service layer has its importance in mediating the upper layer of business metadata and the lower layer of information metadata to construct the coherence of enterprise integration. From an architectural perspective, service can exist in user access, application, information and infrastructure while from a run-time architectural perspective, the user interface service at the upper layer needs a semantic web to make information accessible to human and software agents on a semantic basis [9] [10].

Step 4

Once the validation of business components and their associated service components is completed in steps 1, 2 and 3, we can move on to service provisioning based on the components we selected. The provisioning is an architectural process to produce both common and composite components by means of use case parsing to count the frequency and dependency of service components. Frequency means a count of the appearance of a component in the parsing process; dependency means that two or more components work together frequently and follow a particular sequence in forming a composite service or process.

In the use case parsing process, first the finite Service matrix S_{ij} (i = 1, 2, 3 ... n, j = 1,2,3 ... n) is plotted in both vertical and horizontal dimensions in order to predict both the common degree and the degree of sequential dependencies. We have designated the map produced by the plotting a CCSA (common component and service aggregation) map; the process is called a CCSA process. Each service S, when plotted in the CCSA map, refers to the aggregation of service aggregation. This means that a combination of

service components, if they follow a certain sequence with a certain frequency, may qualify for service aggregation; that is, two or more than two atomic service components can be aggregated upward to a coarse grained service in which the grain size and threshold is determined by the architecture group using either empirical benchmarking or optimization in the evaluation of advantage (i.e., coarse grain helps in minimizing interfaces) and disadvantage (loss of flexibility in the new configuration).

In component evaluation, for their significance in relation to common degree and level of aggregation, let S* denote the finite set of services,

$S^* = \{ S_1, S_2, S_3 \ldots S_n \}$, $n \geq 2$,

Let S^{cc} denotes the finite set of services which qualified as a common component, we derive, $S^{cc} \subseteq S^*$, let $\tilde{v}_k^*$ being the normalized index showing the intensity of common degree

for kth service, k = 1,2... n. let $v_{critical}$ be the threshold in qualifying a common component, the ratio of $\tilde{v}_k^*$ over $v_{critical}$ determines whether S_k is qualified as a common component.

In the evaluation of the significance factor of service component in K-B, B-S and S-C transformation, let

$P(X^k_j)$ be the jth member's weight for kth KPI

$\Pi(X^k_{ji})$ be the jth member's preference of component X_i for kth KPI

$\Phi(X^k_i)$ be the generic significance factor which is a concatenation between the weight of the role and preference of component X_i for kth KPI

$\Phi^*(S^{kb}_i)$ be the business significance factor in the B (business) dimension, which is a concatenation between the weight of role and preference of component S_i for kth K (KPI) and bth B (Business component)

$\Phi^*(C^{kb}_i)$ be the physical component significance factor in the I (information component) dimension, which is a concatenation between the weight of role and preference of component C_i for kth K (KPI) and bth B (Business component)

K-B transformation (size of Business component = n, size of decision group = m)

$\Phi(B^k_b) = \Sigma P(B^k_j)\, \Pi(B^k_{ji})$, let $\mu B^k = Max(B^k_b)$, denote the maximum value for B^k_b
$_1 < b < n$ $\quad _1 < j < m$

$\Phi^*(B^k_b) = \frac{\Phi(B^k_i)}{\mu B^k}$ be the relative value for B^k_b $_{(1 < i < n)}$ when compared with μB^k

B-S transformation (size of Service component = n, size of decision group = m)

$\Phi(S^k_i) = \Sigma P(S^k_j)\, \Pi(S^k_{ji})$, let $\mu S^k = Max(S^k_i)$, denote the maximum value for S^k_i
$_1 < i < n$ $\quad _1 < j < m$

$\Phi^*(S^{kb}_i) = \frac{\Phi(S^k_i)}{\mu S^k} \Phi^*(B^k_b)$ be the relative value for S^k_i when compared with μS^k

S-C transformation (size of Component element = n, size of decision group = m)

$\Phi(C^k_i) = \Sigma P(C^k_j)\, \Pi(C^k_{ji})$, let $\mu C^k = Max(C^k_i)$, denote the maximum value for C^k_i
$_1 < i < n$ $\quad _1 < j < m$

$\Phi^*(C^k_i) = \frac{\Phi(C^k_i)}{\mu C^k} \Phi^*(B^k_b)$ be the relative value for C^k_i when compared with μC^k

S-S transformation (CCSA parsing analysis)

Let $\Psi(S^k_{ij})_{(1 \le i \le n,\ 1 \le j \le n)}$ be the matrix from the result of n-dimension S-S parsing

Let $\gamma(S^k_{ij})$ denote the count of S^k_{ij} (dependency count from S^k_j to S^k_i)

When i = j, $\gamma(S^k_{ij})$ is the statistic of common degree of S^k_{ij}

When i <> j, $\gamma(S^k_{ij})$ is the statistic of dependency degree of S^k_j on S^k_i

$$\Psi(S^k_{ij})_{(1 \le i \le n,\ 1 \le j \le n)} = \begin{bmatrix} \gamma(S^k_{11}), \gamma(S^k_{21}), \ldots\ldots \gamma(S^k_{n1}) \\ \gamma(S^k_{12}), \gamma(S^k_{22}), \ldots\ldots \gamma(S^k_{n2}) \\ \cdots\cdots\cdots\cdots \\ \gamma(S^k_{1n}), \gamma(S^k_{2n}), \ldots\ldots \gamma(S^k_{nn}) \end{bmatrix}$$

Since $\Phi^*(S^k_i)_{(1 < i < n)}$ represents the significant factor of the service component using the political model, and the concatenation of K-B & B-S

$\Phi^*(C^k_i)_{(1 < i < n)}$ represents the significant factor of the service component using a political model, and the concatenation of K-B & S-C.

This implies that the significance of service component S depends on both its associated business component and itself, as does component element C. The reason we use a parsing matrix to represent the significance factor is to try and derive dimensional vectors by means of different patterns.

Step 5

Both $\Phi^*(S^k_i)$ and $\Phi^*(C^k_i)$ statistics are shown in Figure 3 as an example of CCSA parsing analysis associated with atomic service components in provisioning upward service through the alignment of S-C, S-S and B-S transformation. It introduces the structure of the transformation with focus on the S-S step. Aggregation in the service layer is important as it eliminates boundary configuration, increases the integrity of common services in an error-free configuration environment and, furthermore, can be developed into coarse grain processes or processes adjacent to the business layer. Therefore the referential alignment of B-S strategy uses coarse grains to support business components directly; those fine grains at the bottom of the service layer can form a different pattern in supporting the coarse grains. This implies that the fine grains serving as the foundation of the services can be reused so that one service can contribute to provisioning many coarse grains through different patterns. The formation of a pattern can use common components as a base as they are common in provisioning generic services in each business component. Aggregation that is based on a pattern derived from parsing analysis as a usual case is crucial in common services; in the terminology our use of "common" indicates an encapsulated enterprise service that is seldom changed even though a light revision is acceptable.

The information aggregation in the component layer forms the metadata that is stored in an information repository distinct from the service components. It constructs part of the component element in the physical layer. The component element consists of physical units of functional and non-functional elements; while an agent is an example of a functional element, a business object or a security component are both considered as non-functional elements.

Service aggregation is a process for deriving the metadata within a layer, so even though the service components and component elements have been well defined in the previous 1-5 steps, fine tuning is required to derive the final step.

In Step 4 we described how the service component is aggregated. Within the component layer there is no such process; as the physical layer is at an atomic level loosely coupled

with peer components and associated services, we assume all the component elements in information component form are in an optimized state and that no further aggregation is required. Figure 5 illustrates the concept of how the data is aggregated toward the service layer with several steps being required to streamline the data transformation from raw data to abstract information. Therefore, before any information component is defined, a suitable aggregation in optimization is necessary; this implies that the aligned service component is always associated with an optimized component. Thus, if the component is an information component, there is no further aggregation required after S-C is configured.

Each project or enterprise activity has its unique strategy, business context or KPI that first needs to be converted to business components. The alignment can reuse previous experience in a top-down K-B-S-C approach by selecting the same business components, service components and physical components to establish the business goal; however, in most project scenarios only part of the empirical K-B-S-C transformation process can be reused. This leaves a large portion of the alignment as either new material or a deep modification which requires starting from a business component selection.

In the service component-based architecture, interoperability within K-B-S-C is the key in building the skeleton of enterprise integration in a vertical direction. To cope with this objective, we apply both a political model and a rational model to validate the components in each layer. The political model applies group decision-making while the rational model applies parsing analysis; both models work on component validation with different focus of scope.

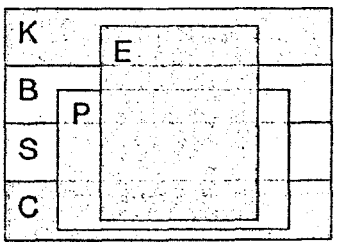
In role-based group decision-making, each role has a different weight in each layer within B-S-C and each group member has a different preference in their component selection. We use the political model, starting with the business components. We repeat the political model in a use case deconstruction to derive service components and physical elements. Inthe following K-B-S-C political model evaluation, a bottom-up approach using rational model parsing is applied to S-S transformation. This step is the last one in K-B-S-C transformation and the objective is to derive common components and service aggregation. The statistics of parsing results construct a matrix of common components and aggregated coarse-grained components which can be a candidate for process; this bottom-up approach attempts to create visibility of common services in either component units, interface or composite components with their priority and dependency. This provides the architects with a roadmap for building up their enterprise integration and strengthening the alignment and interoperability within the K-B-S-C transformation.

type	Conventional development	Component base development
Analysis	Full on-demand domain	Subset of on-demand domain
Architecture	Use-case driven in full domain	Component selection and aggregation
Development	Start from scratch or adds-on	Component reuse or adds-on
Integration	Horizontal system to system	Vertical metadata to horizontal services
Future dev. effort	Med. to high as lack of traceability	Lower as "reuse" and "loosely coupled"
Service level	Low to medium - scale & constraints	Higher - service virtualization

Table 1. Benchmarking of integration methods in conventional and component base

Layer	Denotation	Role	K-B *weight (level1)*	B-S *weight (level2)*	S-C *weight (level3)*	S-S * *(level 4)*
K	Strategy	Stukeholder	R1	R3	R4	n/a
B	Business	Business Manager	R1	R2	R4	n/a
S	Service	Business architect	R3	R2	R3	n/a
C	Component	Appl. & System Architect	R4	R3	R1	n/a
E	Enterprise	Enterprise Architect	R2	R1	R2	n/a
P	Project	Project manager	R3	R2	R2	n/a

* S-C is based on the information aggregation and intensity of data access
S-S applies rationalization model in common component parsing and service aggregation
"weight" denotation – R1: 1.0/ R2: 0.7/ R3: 0.4/ R4: 0.1



Inter-Layer - Denotation	Decision method
K-B - Business component decision	Political model, human decision
B-S - Service component decision	Political model, human analysis
S-C - Information component assessment	Political model, human analysis
S-S - Common Service & Aggregation	Rational model, parsing analysis

Table 2. Weight classification for role and layer

U1	Buyer sends payment information to the merchant server. Merchant software must send and receive messages; encrypt and decrypt; store public and private keys; and request and receive certificates.	E1	S1	Virtual (payment_info)	C11	Buyer_req_merchant_payment (info)
		E2	S1	Virtual (payment_security)	C12	Merchant_exec_buyer_payment (security)
		E3	S2	Online (auth_info)	C21	Merchant_exec_buyer_messg ing
U2	Merchant software takes payment information from the cardholder and sends it to the acquirer (merchant's bank). Acquirer institution must receive and authenticate payment information the merchant received from the cardholder.	E4	S1	Virtual (payment_info)	C13	Merchant_submit_acquirer_pay ment
		E5	S2	Online (auth_tx)	C22	Acquirer_authenticate_merchant_ payment
		E6	S3	Security (merchant)	C31	Acquirer_validate_merchan t_tx
		E7	S1	Virtual (payment_merchant)	C14	Merchant_charge_acquirer_comp uting
U3	Acquirer sends an authorization request to the issuer over the interbank network. Issuer sends the authorization response to acquirer.	E8	S4	Cardholder (status)	C41	Acquirer_req_issuer_auth
		E9	S5	Account (3rd party)	C51	Interbank_reps_acquirer_a uth
		E10	S2	Online (auth_interbank)	C23	Issurer_respond_interbank _auth
		E11	S3	Security (cardholder)	C32	Interbank_check_cardholder_f raud
U4	Acquirer notifies merchant about the status of authorization; if the response is positive, merchant fulfills the order.	E12	S4	Cardholder (status)	C42	Acquirer_notify_merchant_st atus
		E13	S5	Account (merchant)	C52	Merchant_check_acquirer_sta tus
		E14	S6	Recur (merchant)	C61	Merchant_fulfill_acquirer_o rder
U5	Merchant presents the charge to the acquirer bank.	E15	S3	Security (merchant)	C33	Merchant_charge_acquirer_v erify
		E16	S1	Virtual (payment_merchant)	C15	Merchant_charge_acquirer_comp uting
U6	Acquirer sends a settlement request to the issuer.	E17	S7	Settlement (batch_run)	C71	Acquirer_run_issuer_settle ment
		E18	S7	Settlement (batch_report)	C72	Acquirer_send_issuer_settle ment
U7	Issuer charges the buyer's credit card account and at regular intervals notifies buyer of the transactions and accumulated charges.	E19	S1	Virtual (payment_issuer)	C16	Issuer_charge_by_interban k
		E20	S4	Cardholder (status)	C43	Issuer_run_report_buyer (accumulated)
U8	Buyer pays the charges to the bank. Acquirer credits merchant's account.	E21	S5	Account (buyer)	C53	Buyer_pay_issuer_interbank (credit card)
		E22	S5	Account (acquirer)	C54	Issuer_credit_acquirer_buy er
		E23	S5	Account (merchant)	C55	Acquirer_pay_merchant_issurer (buyer)

Table 3. Component mapping between each layer

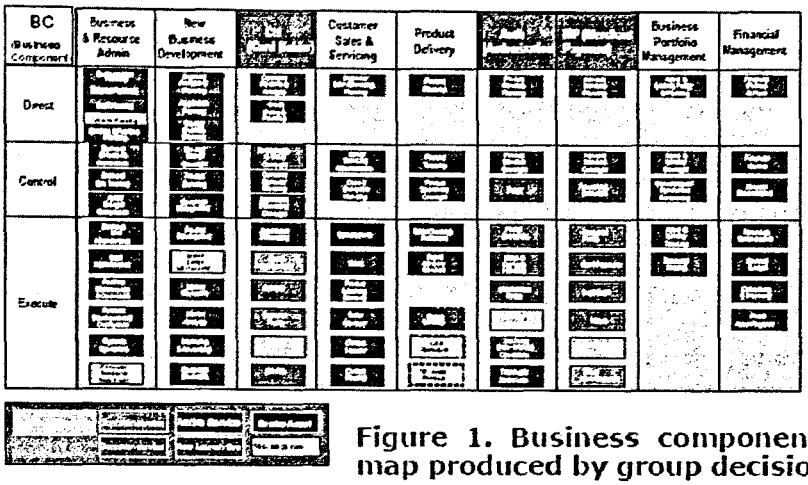


Figure 1. Business component active map produced by group decision

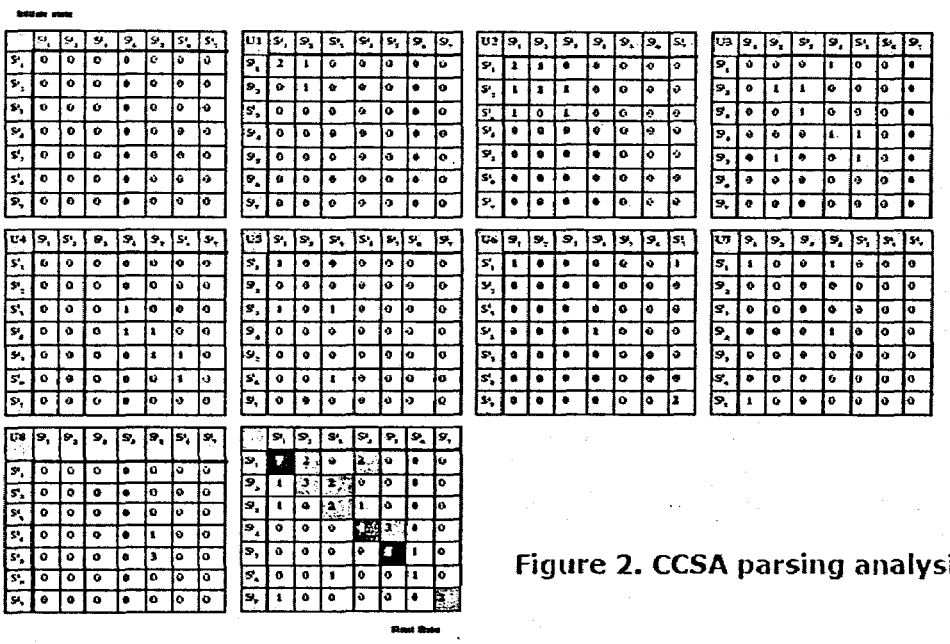


Figure 2. CCSA parsing analysis

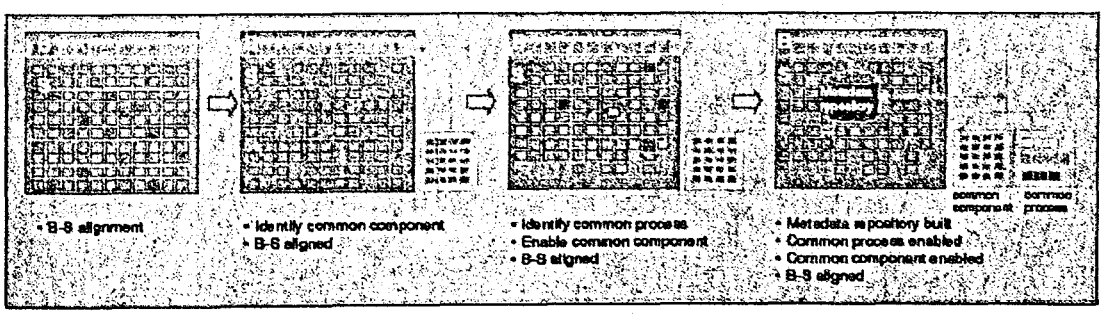


Figure 3. Transformation in constructing common component and common service

We thank reader in reading our publication, specials thanks to The 11th International Conference on Human Aspects of Advanced Manufacturing Agility and Hybrid Automation in publishing our research work.

1. Raymond Wu., (2006), "Enterprise Integration in semantic metadata", IEEE (IT professional) ITPro-0055-0806
2. Jie Lu, Guangquan Zhang & Fengjie Wu, (2005), "Web-based Multi-Criteria Group Decision Support System with Linguistic Term Processing Function", IEEE Intelligent Informatics Bulletin June 2005 Vol.5 No.1
3. Remko Scha, Rens Bod[5] and Khalil Sima'an, (1999), "A Memory-Based Model of Syntactic Analysis: Data-Oriented Parsing" Special Issue on Memory-Based, Language Processing) 11, pp. 409-440.
4. Raymond Wu, (2006), "Enterprise Vertical Integration in the e-Government", Electronic Government, an International Journal (EG) ISSN (Online): 1740-7508 - ISSN (Print): 1740-7494
5. S. Dwarkadas K. Gharachorloo L. Kontothanassis D. Scales M. L.

   Scott R. Stets, (1998) ," Comparative Evaluation of Fine- and Coarse-Grain Software

   Distributed Shared Memory", Cambridge Research Laboratory
6. Curtis, K., Foster, P. W., Stentiford, F. (1999), "Metadata - The Key to Content Management Services", Proceedings of the Meta-Data '99, The third IEEE Meta-Data conference, April 6-7, 1999, Bethesa, Maryland, USA
7. David Wang, (2000), "Automated Semantic Correlation between

   Multiple Schema for Information Exchange", MIT Sloan School of Management.
8. Traugott Koch, UKOLN, (2006), "Electronic thesis and dissertation services: semantic interoperability", E-Thesis Workshop, Amsterdam 2006-01-19/20
9. M. Staab, N. Stojanovic, R. Studer, and Y. Sure, (2001), "Semantic Portal - The SEAL approach. to appear in: Creating the Semantic Web.", MIT Press
10. O. Valkeapää and E. Hyvönen, (2006), "Semantic Annotation with Browser-based Annotation Tool", 1st Asian Semantic Web Conference

E3

E3: Validation process and component analysis in semantic mediation

Publication: ICIS /IEEE 2007 (http://acis.cps.cmich.edu:8080/ICIS2007)
July 11-13, 2007, Melbourne, Australia
Publication Contact: Professor Dale Jarman
Roger Lee & Morshed Univ.
Jarman@cmich.edu

IEEE COMPUTER SOCIETY
IEEE
6th IEEE International Conference on Computer and Information Science (ICIS 2007)

From: Dale.Jarman@cmich.edu Add to Address Book Add Mobile Alert
To: rwu_academic@yahoo.com
CC: Dale.Jarman@cmich.edu
Subject: Your ICIS 2007 Paper (Number 6)
Date: Tue, 27 Mar 2007 20:22:35 -0500 (EST)

Dear Mr. Raymond Wu:

On behalf of the ICIS 2007 Program Committee, I am delighted to inform you that the following paper has been accepted to appear at the conference:

The Validation Process and Component Analysis of Enterprise Integration

Best Regards,
Roger Lee and Morshed U. Chowdhury
ICIS 2007

Reviewer's Scores

Technical Content: 8
Originality: 8
Clarity: 5
Significance: 6

IEEE computer society

Enter Search Query

Home | Digital Library | Site Map | Store | Contact Us | Press Room | Shopping Cart | Help | Login |

digital library

DIGITAL LIBRARY HOME
BROWSE BY TITLE
BROWSE BY SUBJECT
SEARCH
LIBRARY/INSTITUTION RESOURCES
RESOURCES
SUBSCRIPTION
ABOUT THE DIGITAL LIBRARY

Published Articles >> Table of Contents >> Abstract

6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007) pp. 873-878

Validation process and component analysis in semantic mediation

Raymond Wu, IBM Software Group

Full Article Text:

DOI Bookmark: http://doi.ieeecomputersociety.org/10.1109/ICIS.2007.193

Send link to a friend

Abstract

Both of decision input and element analysis play crucial roles in component-based design. In the past decade, much of the validation process have been massively implemented by group decision. In this research, we take the output from group decision and extend the focus of the methods and techniques to component analysis by using semantic mediation techniques in supporting the enterprise integration and ontology interoperability. We believe, both human interactive and semantic mediation analysis follow the nature of business in pre-design stage, and semantic mediation in supporting component analysis is crucial in early stage of enterprise integration. Our proposal mainly argues that semantic mediation need to impact components first and push semantics strategy directly into the micro process to be synthesized into the service layer and this is the main objective of component analysis.

Additional Information

Back to Top

Citation: Raymond Wu, "Validation process and component analysis in semantic mediation," icis, pp. 873-878, 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007), 2007

Published Articles >> Table of Contents >> Abstract

6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007) pp. 873-878

Validation process and component analysis in semantic mediation

Raymond Wu, IBM Software Group

URL :http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/proceedings/&toc=comp/proceedings/icis/2007/2841/00/2841toc.xml&DOI=10.1109/ICIS.2007.193

Author Index

Shudong, Shi 153
Sibai, Fadi N. 604
Siddhurth, N. 1049
Sinnott, Richard 906
Woods, Megan 412, 581
Wu, Gin-Der 43, 315
Wu, Hon Ren 425
Wu, Raymond 873

6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007)

Validation process and component analysis in semantic mediation

Raymond Wu
IBM Software Group
raymond@de.ibm.com

Abstract

Both of decision input and element analysis play crucial roles in component-based design. In the past decade, much of the validation process have been massively implemented by group decision. In this research, we take the output from group decision and extend the focus of the methods and techniques to component analysis by using semantic mediation techniques in supporting the enterprise integration and ontology interoperability. We believe, both human interactive and semantic mediation analysis follow the nature of business in pre-design stage, and semantic mediation in supporting component analysis is crucial in early stage of enterprise integration. Our proposal mainly argues that semantic mediation need to impact components first and push semantics strategy directly into the micro process to be synthesized into the service layer and this is the main objective of component analysis.

Introduction

We continue the previous strategic work in enterprise integration, with further investigation of the human factors in interoperability within enterprise integration. Our enterprise integration in vertical direction is a metadata driven approach in the component-based environment, the early stage involved in decision process in following three steps: i) Validation process of business components ii) Validation process of service components iii) Validation process of physical components. We start from the decision-making process of business component selection to derive the service components and physical elements. The decision-making process uses group decision to take each individual opinion or preference and multiply it by a factor [1]. The decision makers are motivated to communicate and share opinions within the group in making business component selection. We use the same political model in all business, service and physical component validation and this is what we called top down validation for atomic elements which is part of political model. Followed by validation process, we proposed the framework in keeping alignment between these different components of different layers in building up their interoperability and the semantic strategy between fine grained component and further development of coarse grained service component is called semantic mediation in component analysis which is part of rational model. Both political and rational models are constructed in a component-based environment; this is because in a scrambled development environment provides no opportunity for group decision-making or rational analysis, in the other words the impact from decision and targeted scope are mismatch, so the decision and analysis cannot be effectively deployed into the implementation, thus the impact is high while the interoperability is low. In conventional development, weak visibility between business, service and information makes the linkage and enterprise interoperability impossible. Once the alignment is missing or invisible between layers, the common services and service virtualization become impossible as these require a foundation in both macro interoperability and micro mechanism. This results in a silo system with serious limitation of services and weakness in enterprise integration [2], as a matter of fact, enterprise integration gain no benefit from group decision and deviate from the nature of business. As mentioned above, the methodology we used in this research work included the component model, political model, and macro metadata integration; we differentiate these with conventional methods in the criteria listed in Table 1.

Before we proceed to the key area of these steps, architecture terminology "K-B-S-C transformation" has been used to represent interoperability between architecture layers. We began with enterprise strategy (K), the primary area of concern for stakeholders and business managers, to derive the components in both business (B) and services (S) layers. As each layer of K (context), B (conceptual) and S (logical) are all at the conceptual or logical level, the bottom layer which is the physical component (C) forms a loosely-coupled alignment with S in provisioning bi-directional interoperability. The micro mechanism between these two layers

forms the service unit; aggregate individual service provides service in the business layer to finally achieve the strategic goal.

Literature Review

Lagoze and Hunter (2001) build a conceptual model to facilitate interoperability between metadata ontologies from different domains. Their model uses Entity as the root class and assigns three categories Temporality, Actuality, and Abstraction as its subclasses. This imply the communication between domain ontology is possible, however the business decomposition was not covered in the literature. In our ontology approach, business component grouping and validation are all the strategy and initial steps in establish the component based service provision in vertical direction.

Semantic heterogeneity and mediation has been investigated in various literatures (Naiman and Ouskel, 1995; Goh, 1997; Ram and Park, 2004) [3] [4] without a consensus within their research. In enterprise integration, a common view of semantic interoperability is this it is involved in the degree of flexibility two different metadata sets have in working together and understanding each other using a common terminology. Apart from this peer-to-peer concept, semantic interoperability has special meaning in our research, indicating the strategy and capability of cross layer transformation which takes business semantics into component-based implementation. Thus the atomic element, in the form of a service component, "inherits" the micro semantic and syntax in mediating the information element which is in metadata form.

From our observation and project experiences, our motivation for the development of semantic technologies did not originate from a conventional "divide-and-conquer" mindset, which merely takes the output from business analysis and interpretation for further development. We saw that some crucial points were missing in our literature review and consider this crucial because within enterprise, the skeleton of dimensional service with linkage between business, service, component and information is mandatory. By means of these dimensional services, enterprise and industry have the opportunity to provision their strategic dimensions with further impact on patterns and common components.

Validation in component based environment

Validation is mainly based on the assessment of business criteria via human decision-making with the scope of validation covering each layer of business, service and component element. There are two factors influence the validation process. First, group members, who involved in validation, have different "weights" in a group decision-making process; this fact should be taken into account in the generation process of the satisfactory group decision. The second factor is the individual's preference for specific components. The final decision on a business component and its associated service elements needs vision from corporate strategy, business goal and enterprise architecture in order to ensure that all the different viewpoints shed light on a common focus. Once business components are selected for a project with alignment to K, the derived services from B will be assessed and validated. The semantic metadata within K-B-S-C therefore provides a means to hold their matrix alignment; for design purposes, the referential integrity and semantic interoperability require that there be a repository to hold the information. The K-B-S-C metadata integration forms a skeleton in enterprise integration with the choreographic capability of retrieving each layer's metadata for the details of each component stored in the metadata repository of B, S and C. This macro to micro hierarchical structure constructs a vertical interoperability skeleton in which the different roles work together; the architect takes care of the main route of the skeleton in metadata integration and the developer takes over the skeleton's work and extends the micro design in the transformation of components and interconnectivity of service. Service provisioning is based on an assessment of frequency of use of a component and its dependency within a project. Before implementation, the strategic KPI will first be reviewed based on a project's requirements followed by business component selection and validation. Once the business components have been validated, the deconstructed service components and associated component elements will be evaluated and validated, using the same validation methodology [5]. Figure 1 shows the validation process which derived from business component. Once a business component is selected, the associated service components can be derived. The weight classification of the role and preference is the same as above but the weight granted to each role is different. This is because when the focus is

shifted from K-B to B-S, the more strongly-weighted role will be shifted from stakeholder to architect. Before input is gathered from each individual, the architect team (mainly the enterprise and business architects) will describe each service component to the group so that each member has a knowledge base from which to express and to determine the significance of a service component from an enterprise point of view.

Service mediation

The service layer has its importance in mediating the upper layer of business metadata and the lower layer of information metadata to construct the coherence of enterprise integration. From an architectural perspective, service can exist in user access, application, information and infrastructure while from a run-time architectural perspective, the user interface service at the upper layer needs a semantic web to make information accessible to human and software agents on a semantic basis [6] [7]. Service components, derived from the business use case, are the mediation between business and technical elements; they integrate all business semantics, interfaces and technical configurations into a logical component. Each service component is initially at an atomic level, meaning that it consists merely of nouns and verb from a single subject, verb and object in business terms. In other words, service components guide technical implementation for the interface and provide metadata for physical deployment.

Beneath this metadata mapping there is other information to specify the details of each use case, service component and component element with regard to their interface, semantic notation, business logic, data access and component configuration.

Producing a Service pattern through component evaluation

Our construction of metadata mediation is based on component analysis and dimensional service. The initial component analysis in an enterprise should be merely through decomposition and aggregation, so when a business goal is aligned with dimensional services, each dimension can be a composite service which consists of many components within different service categories. These dimensions need patterns to fulfill their technical implementation and, as the micro process in the service pattern needs a map to derive the dimension, the service map needs to be the discipline of pattern design and serves as a modeling tool for dimensional service. To demonstrate how the service map is created, let $\Phi(C_n)$ denote the matrix of Cartesian space from the component comparison for component analysis; where θ_{ij} is the comparison between adjacent components C_j against C_i

$$\Phi(C_n) = \begin{bmatrix} \theta_{11}\ \theta_{12} \ldots . \theta_{1n} \\ \cdots \\ \theta_{n1}\ \theta_{n2} \ldots . \theta_{nn} \end{bmatrix}$$

From the eigenvalue base equation, $AX = \lambda X$ where X is the eigenvector and λ is the eigenvalue, we derive the equation which follows.

$$\begin{bmatrix} a_{11}-\lambda & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22}-\lambda & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kk}-\lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Figure 2 shows the evaluation of dimensional service which is based on the best practice principle of semantic mediation, and the comparison of pair-wise components against a service dimension is based on the relative importance of one component versus another on the same level. The evaluation shows how pair-wise comparison derives the relative weight of each component in relation to a service and how the service component is composed from component elements with different weights; these provision a dimensional service with their individual weight. In our evaluation we derived three level iterations: 1) how the component mediates to achieve a service component; 2) how the service map is created through service components; and 3) how the dimensions achieve a business goal.

Therefore, the determination of the service atomic component is the key for moving component-based architecture into semantic service. We are unable to merely rely on the configuration of the component in achieving dimensional service, as it does not make too much sense in semantic mediation within enterprise integration. However, we certainly need to apply component-centric design to fulfill interoperability requirements. As an enriched services layer produces coarse-grained services which are aggregated from fine-grained components. Service dimensions, therefore, become an easily

configured soft area for the provision of a full range of services. The notable strategies and characteristics of interoperability include: a universal schema and metadata in the information layer; semantic mediation between business, service and component; interoperability between three repositories; and business decomposition in the upper layer. Among these, the determination of the service atomic component is the key for moving component-based architecture into semantic service. We are unable to merely rely on the configuration of the component in achieving dimensional service, as it does not make too much sense in semantic mediation within enterprise integration. However, we certainly need to apply component-centric design to fulfill interoperability requirements. The reasons for this are: 1) The service atomic component is the mediation unit between business and IT as this atomic element loosely binds business and IT terminology in a common view; only through this single point design do we have the opportunity to align enterprise architecture into interoperability; 2) From this single point design we can validate associated metadata in the repository for their analysis in noun, verb and attribute as an atomic element while using business semantic terms to mediate pre-built component sets which have been configured for that service; 3) The service component can be easily built, revised or aggregated to achieve any service change caused by business change because most of the physical component design is an atomic level function. The mechanism of interfacing and parameter transfer are all keys to how each component performs; therefore, the mechanism of the "integrated functions" should not be vaporized, rather they need to be stable and durable. This means the virtual service which is mounted on the supporting components should effectively apply metadata in the repository to mediate other services in provisioning a service dimension; once this service is validated, the metadata which holds the mediation information should be stored in a knowledge-based repository for reuse or revision; 4) The service layer is an easily configurable area which we can leverage to the scenarios of cache memory computing in mediating application and database storage. The syntax, logic, semantic and execution plans such as security features are all under metadata control; these are stored in repository and ready for configuration.

Conclusion

We repeat the political model in a use case deconstruction to derive service components and physical elements. The group members validate both service components and component elements by using the same method with differently weighted roles. The evaluation between service components and physical components is based on how significant their contribution to the business goal is from a logical and physical information component perspective; a higher degree of information aggregation of C within B-S-C gives a higher preference weight in that physical component. The multiplication of B and S significant factors represents the significance of S within the K-B-S-C transformation; using the same method, the multiplication of B and C significant factors represents the significance of C within the transformation. The objective of K-B-S-C transformation and significance factor evaluation is to try to build up the vertical enterprise skeleton and extend the key services to their peripheral surroundings in performing full service virtualization. In role-based group decision-making, each role has a different weight in each architecture layer and each group member has a different preference in their component selection. We use the combination of these two factors in political model to validate the component against dimension.

The validation process support the succeeding component analysis, and as service interoperability which is our goal, doesn't rely merely on conventional integration. Semantic mediation is mandatory to synthesize K-B-S-C in micro process so each component has opportunity to interoperate toward the enterprise goal. Currently, there are two significant issues which need immediate action: first, enterprise misalignment which, as we pointed out in previous research, indicates that an enterprise loses governance in semantic interoperability in the maintenance of coherence between the business, service and technical components; and second, semantic mediation, which has been misinterpreted by several activities and needs a new direction in order to fix the following common issues: 1) research has been limited to data layer with constraints in its ontology; 2) the scope of interoperability has not been stretched out to the global scale, thus the mediation has been limited to only conjunctions of the architecture layer such as application interface, system protocol, and metadata messaging. We believe the ontology needs a robust

mechanism in architecture; this will require a "big picture" that encompasses the above areas into interoperability and is what we mean by the "skeleton" of common service.

In the service component-based architecture, interoperability within K-B-S-C is the key in building the skeleton of enterprise integration in a vertical direction. To cope with this objective, we apply both a political model and a rational model to validate the components in each layer. The political model applies group decision-making while the rational model applies parsing analysis which will be introduced in a separate literature; both models work on component validation with different focus of scope. Political model is the foundation of rational model as the nature of business implementation started from group decision which cannot be replaced by machine or other rules, without the human effort and validation, all the analysis and implementation will be in vain.

Stage	Conventional	Component based
Domain Analysis	On-demand full domain	Subset of domain analysis
Enterprise Architect	Use-case driven through full domain	Component based driven and
Development	End to end development from biz to data	Component selection in a small scope
Integration	Gap exist from domain, timing and project	Alignment in both vertical and horizontal are clear
Development effort	High effort in new dev. No reuse is available	High effort in new dev. Reuse is available
Risk	High, as scoping is difficult	Low as scope is clear
Maintenance effort	High, any change cause big effort	Low as each change can be identified and reused
Service level	Low to medium with constraints	High, service virtualization

Table 1. Benchmarking of integration methods in conventional and component base

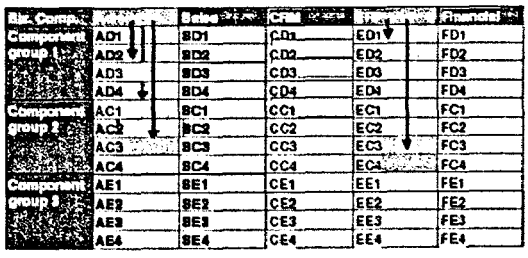


Figure 1. Business decomposition in forming a pattern

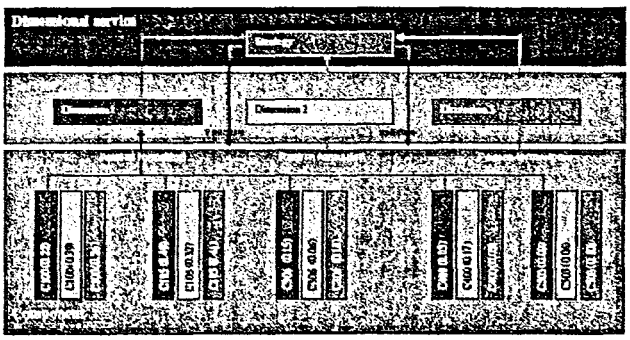


Figure 2. Component evaluation in dimensional service

Reference

1. Jie Lu, Guangquan Zhang & Fengjie Wu, (2005). "Web-based Multi-Criteria Group Decision Support System with Linguistic Term Processing Function", IEEE Intelligent Informatics Bulletin June 2005 Vol.5 No.1
2. Raymond Wu, (2006), "Enterprise Vertical Integration in the e-Government", Electronic Government, an International Journal (EG) ISSN (Online): 1740-7508 - ISSN (Print): 1740-7494
3. Kashyap V. and Sheth A. (1996), "Semantic and Schematic Similarities between Database Objects: A Context-based approach", VLDB Journal: Very Large Data Bases
4. Sergey Melnik, (1999), "Generic Interoperability Framework", Distributed and Parallel Databases Journal 1999
5. Mishra A., Ripley M., and Gupta A. (2002), "Using Structural Analysis to Mediate XML Semantic Interoperability", MIT Sloan Working PP 4345-02 No. 1024-05 February 2002
6. M. Staab, N. Stojanovic, R. Studer, and Y. Sure, (2001), "Semantic Portal - The SEAL approach. to appear in: Creating the Semantic Web.", MIT Press
7. O.Valkeapää and E. Hyvönen, (2006), "Semantic Annotation with Browser-based Annotation Tool", 1st Asian Semantic Web Conference

E4

E4: Determination of Common Component Integration

Publication: ACM NZ
ISSN 1176-9998 (http://www.acm.org/chapters/acm_nz/bulletin/vol3/issue1)
2007 Volume 3 Issue 1
http://www.acm.org/chapters/acm_nz/bulletin/vol3/issue1/
Publication Contact: Taiyu Lin, Inf. Systems
Department, Massey University,
New Zealand t.lin@massey.ac.nz



Date:	Sat, 19 May 2007 13:25:39 +1200
From:	"Taiyu Lin" <taiyu.lin@gmail.com> Add to Address Book Add Mobile Alert Yahoo! DomainKeys has confirmed that this message was sent by gmail.com. Learn more
To:	"Raymond Wu" <rwu_academic@yahoo.com>
Subject:	Re: acm_nz: submissions

Dear Raymond,

Yes, manuscript 2-5 (Determination of Common Component Integration) i already accepted and published. See
http://www.acm.org/chapters/acm_nz/bulletin/vol3/issue1/
You can open the printable version PDF to find out more details (e.g. page numbers). I have already sent you a notice about this, maybe tha

Best regards,
Taiyu Lin
Editor, ACM_NZ Bulletin
http://www.acm.org/chapters/acm_nz/

Wu, R. (2007). Determination of Common Component Integration. Computing Technology New Zealand Bulletin, 3(1), 23-35. (ISSN 1173-0986)

Determination of Common Component Integration

Raymond Wu
IBM Software Group
rwu_academic@yahoo.com

Abstract: A study in the importance of componentization, method of creating visibility and the development cycle of common components was conducted. Componentization is the first step in component-based design which aims at enterprise interoperability. Failure of interoperability is one of the top IT issues in this decade and results in enterprise misalignment and development silos. However in a component-based environment, enterprise interoperability can only be achieved by effective identification of common components for further development. The common component concept proposes that any business, service or technical components can become common in the industry if they are well defined, designed and aligned. This concept makes feasible standardization and reuse: the more we deploy common components in a prebuilt foundation, the higher compliance within the industry we will achieve. This strategy implies that common components will shorten the development lifecycle, improve quality and eliminate integration boundaries, and furthermore can facilitate common services across enterprise and geographical limitations into the industry main stream if further development can be provided.

Keywords: Enterprise, Integration, Common component, Interoperability, Service base

Introduction

Common service supported by common components has often been used in the use-case driven process. In the customer-based service industry, an "account status query" can be easily identified in many requests or processes. Once we can align this common service with both physical and business components for their unique alignment, it can be called a common component.

Common components can encapsulate both conceptual and physical elements in provisioning services. A common component is not limited to a functional element; any information or system-driven element such as business object, portal UI element, data access protocol or component interface would qualify as a common component candidate. The common component is the foundation of interoperability in creating a homogeneous environment for enterprise integration.

Enterprise interoperability is the architectural coherence between micro component and macro integration that builds on the foundation of a validation process of decision, design and provisioning stages. Common service is a conceptual term with alignment to all business and service components and physical objects; these can be either functional or non-functional elements. An "account balance query" may consist of a combination of a "multi-account identified" functional component and a "database query" non-functional component.

Before we move on to the core area of this study, we would like to provide a clearer explanation of the terminologies we use. We term enterprise vertical alignment as an enterprise integration approach that creates vertical alignment between business, service and physical components. As in component-based design, componentization, component analysis and validation, vertical alignment and service provision are all key processes in enterprise integration. We will discuss more on how the concept of vertical alignment fits into the enterprise skeleton.

The enterprise skeleton consists of those components which are frequently used, business critical or aggregated from atomic-level elements. When they are identified as common, these components need to be precisely designed to mediate semantics between business and service layers [1]. The techniques used in producing common components are mainly supported by parsing evaluation and metadata repository. Parsing is an analysis process which takes the business use case and deconstructs it into atomic elements in all business, service and physical layers. As each component element is labeled with an ID, parsing evaluation reads elements and their movement like an Automata Tuning machine. The "state" concept records the "string" it reads, which consists of elements and their dependencies, and produces both knowledge and intelligence-based component analysis. The parsing results, which are held in metadata repository, will identify common components and aggregated components in constructing the enterprise skeleton.

The metadata repository also contributes the vertical alignment in forming the skeleton of enterprise, and this provides opportunity for the enterprise to stretch out the skeleton to project-based components. This increase reusability, eliminates the integration boundary and strengthens enterprise integration from an interoperability perspective [2]. Once the visibility of enterprise vertical alignment is created, the metadata of a component-based configuration forms a clear picture of the enterprise skeleton using metadata integration between business, service and information. Through a metadata approach, we can either reuse the metadata mapping and configuration from successful experience or construct a new map for the project by using the configuration of common components and service aggregation as a project portfolio while extending the implementation to configuration of less significant components. Moreover, once the service components and aggregation services are built they can either be reused or aggregated to a coarser grain later if the aggregation is common and optimized in the parsing evaluation [3]. These processes produce metadata interoperability for enterprise integration [4] [5].

This study proposes the common component as an approach in achieving enterprise interoperability. In the introduction, it started with enterprise integration, issue identification, component-based design and setting enterprise interoperability as the objective. In discussion of determination of common components, it proposes component identification and validation through group decision, parsing iteration process and metadata repository.

Literature review

According to L. Yu et al. [6], an obstacle to software reuse is the large number of major modifications that frequently have to be made as a consequence of dependencies within the reused software components. Component-based implementation is the foundation in coping with this bottleneck.

Table 1. Component classification

Common module	A business common task unit, no need to break down
Interface	A calling point to other process/terminator in finite state
Comm/Ind. task	A common or individual task, no need to break down
Report & query	Common or individual report or query screen in finite state
Registry	Meta definition for virtual layer, logic and flow
Common utility	A common system utility unit, no need to break down
Common screen	A common UI screen in finite state
Comm/Ind. event	A common or individual system-level task, no need to break down
Repository	Meta definition of code, DB, logic and parameter

The determination of common services and composite services is a use-case driven parsing iteration process; the purpose of the parsing iteration process is to discover the common areas and dependencies between services. The determination of common components and service aggregation, as mentioned above, is through the parsing iteration process by using a use-case driven model. The derived component is classified and shown in Table 1 and Figure 1.

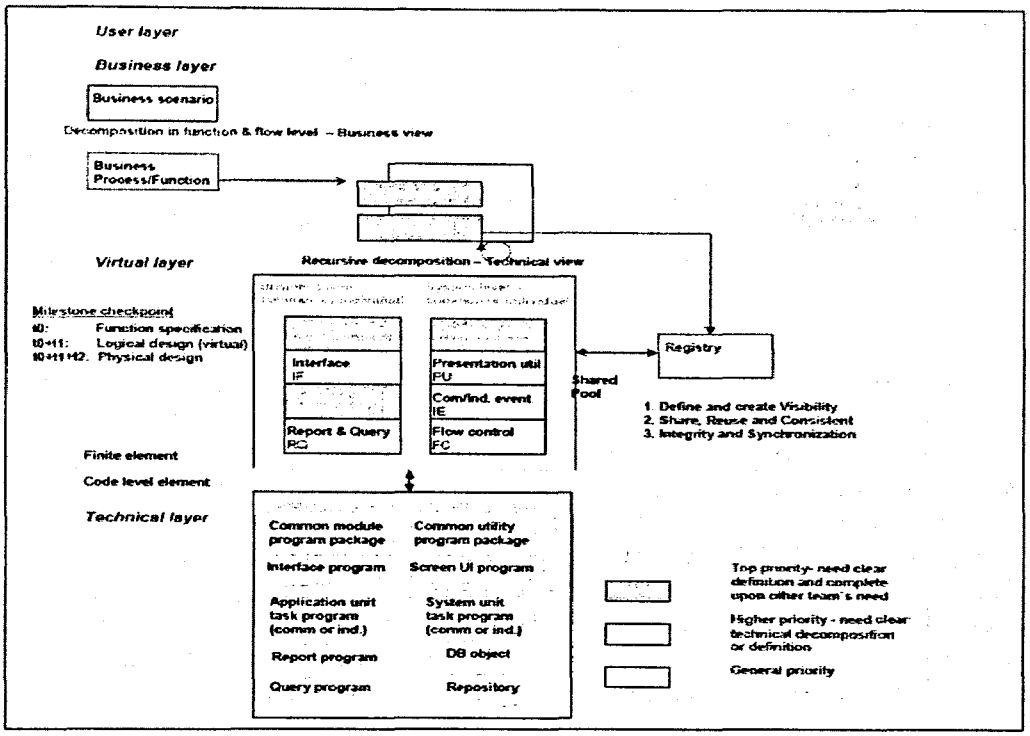


Figure 1. Component classification in service architecture

Determination of common components

The terminology of enterprise interoperability is the key concept for integrating various kinds of information stored in a variety of repositories [7]. In practice, however, interoperability is absent in enterprise integration due to incompatible software, inadequate data formats, semantic heterogeneity and data models [8]. The importance of enterprise interoperability is illustrated in the example of an "account balance query," which may consist of a combination of "multi-account identified" functional components and "database query" non-functional components. The parsing iteration process identifies these two service components as candidates for common service; they are validated once the frequency of the combination passes the appropriate threshold and their interconnectivity design is seen as feasible. Service metadata holds the mapping and routing intelligence that is stored in a service repository so that both common components and the service aggregation have an opportunity to form the main route of the common service skeleton. Any business semantic translation to the process and infrastructure level becomes feasible and efficient under this skeleton [9]. This process is called service transformation.

We would now like to review the lifecycle and positioning service transformation in this roadmap. The group-decision political model is the method used in producing component identification and validation. The group consists of stack holder, business manager and business architect to take care of business components; enterprise architect, process and

information architects to focus on service components; and system and data architects who are responsible for physical components. Once the validation of business components and their associated service components is completed, we can move on to service provisioning based on the components we selected. The provisioning is an architectural process to produce both common and composite components by means of use-case parsing, which counts the frequency and dependency of service components. Frequency means a count of the appearance of a component in the parsing iteration process; dependency means that two or more components work together frequently and follow a particular sequence in forming a composite service or process.

When the architects check the results from the parsing iteration process in the service layer and identify that the frequency, $F(\Phi_i, i-1)$, if $F(\Phi_i, i-1)$, is higher than threshold σ, then from the business perspective the service is common and also business critical in daily operation. If a set of common components follows a certain sequence in each appearance and aggregate into a coarse-grained component, we call this an aggregated component. The aggregation service is the key within business and is crucial from a service perspective as fine tuning becomes possible; we call this the "main route" of the enterprise skeleton. Virtual coarse-grained service eliminates the boundary and leverages many fine-grained components which can be easily configured by adding or modifying the component dynamically to achieve the broader coverage of main route services. For this reason the main route is one of the crucial services in enterprise integration. One example is in the telecommunication industry, where an "open service account", an "open billing account" and "provisioning options for a new customer" are all integrated. This constitutes a main route for "new customer service" coarse-grained service, though it is aligned with fine-grained components to support customer preference, service criteria and billing options which are branches within this main route.

The iteration through each use case will be tracked for its path and service statistics so that once we parse all use cases we should be able to see the frequency of use of each service as well as the path and dependency between services. Where there is a high frequency of services we use the term "common services" and focus on the potential for reuse and minimization of the integration interface. In the case of service paths in which several services have a common sequence and appear in use cases a certain number of times past the threshold, the services can be aggregated to a composite service or potential process. Service aggregation means a composite component consisting of at least two components that form a unique sequence of execution order with a common interface within their boundaries; this composite component will be treated as a coarse-grained service in provisioning the upward aggregation of services. Component evaluation strategy is shown in Figure 1.

The micro mechanism between these each layers forms the service unit; aggregate individual service provides service in the business layer to finally achieve the strategic goal. The denotation is summarized as follows:

K: KPI - strategic Key Performance Identifier and all concerns of the stakeholder

B: Business Component - Unique Business Block - each block may perform individual and independent business

S: Service element - logical service component in the atomic level

C: Physical component associated with service component

Enterprise has its unique strategy, business context or KPI that first needs to be converted to business components. The alignment can reuse previous experience in a top-down K-B-S-C approach by selecting the same business components, service components and physical components to establish the business goal; however, in most project scenarios only part of the empirical K-B-S-C transformation process can be reused. This leaves a large portion of the alignment as either new material or a deep modification which requires starting from a business component selection. The initiative of enterprise integration strategy has three key tasks to perform: service validation, service transformation and service provisioning. Validation is mainly based on the assessment of business criteria via human decision-making; the scope of validation covers each layer of B (business), S (service) and C (component element), so once business components are selected for a project with alignment to K, the derived services from B will be assessed and validated.

In the use-case parsing iteration process, first the finite Service matrix S_{ij} (i = 1, 2, 3 ... n, j = 1,2,3 ... n) is plotted in both vertical and horizontal dimensions in order to predict both the common degree and the degree of sequential dependencies. We have designated the map produced by plotting a CCSA (common component and service aggregation) map; the process is called a CCSA process. Each service S, when plotted in the CCSA map, refers to the aggregation of service aggregation. This means that a combination of service components, if they follow a certain sequence with a certain frequency, may qualify for service aggregation; that is, two or more atomic service components can be aggregated upward to a coarse-grained service in which the grain size and threshold is determined by the architecture group using either empirical benchmarking or optimization in the evaluation of advantage (i.e., coarse grain helps in minimizing interfaces) and disadvantage (loss of flexibility in the new configuration).

In component evaluation, for their significance in relation to common degree and level of aggregation, let S* denote the finite set of services,

$S^* = \{ S_1, S_2, S_3 \ldots S_n \}, n \geq 2,$

Let S^{cc} denote the finite set of services which qualify as a common component, we derive, $S^{cc} \subseteq S^*$, let $\tilde{v}_k^*$, being the normalized index showing the intensity of common degree for kth service, $k = 1,2 \ldots n$, let $v_{threshold}$ be the threshold in qualifying a common component, the ratio of $\tilde{v}_k^*$ over $v_{threshold}$ determines whether S_k is qualified as a common component.

In CCSA parsing analysis, let $\Psi(S^k_{ij})$ $(1 \leq i \leq n, 1 \leq j \leq n)$ be the matrix from the result of n-dimension S-S parsing

Let $\gamma(S^k_{ij})$ denote the count of S^k_{ij} (dependency count from S^k_j to S^k_i)

When i = j, $\gamma(S^k_{ij})$ is the statistic of common degree of S^k_{ij}

When $i <> j$. $\gamma(S^k_{ij})$ is the statistic of dependency degree of S^k_j on S^k_i

$$\Psi(S^k_{ij}) = \underset{(1 \le i \le n,\ 1 \le j \le n)}{} \begin{bmatrix} \gamma(S^k_{11}), \gamma(S^k_{12}), \ldots \gamma(S^k_{n1}) \\ \gamma(S^k_{12}), \gamma(S^k_{22}), \ldots \gamma(S^k_{n2}) \\ \sim\ \sim\ \sim\ \sim\ \sim\ \sim\ \sim\ \sim\ \sim \\ \gamma(S^k_{1n}), \gamma(S^k_{2n}), \ldots \gamma(S^k_{nn}) \end{bmatrix}$$

Since $\Phi^*(S^k_i)_{(1 < i < n)}$ represents the significant factor of the service component using the political model, and the concatenation of K-B & B-S $\Phi^*(C^k_i)_{(1 < i < n)}$ represents the significant factor of the service component using a political model, and the concatenation of K-B & S-C. Figure 2 shows the parsing techniques in producing common component. This implies that the significance of service component S depends on both its associated business component and itself, as does component element C.

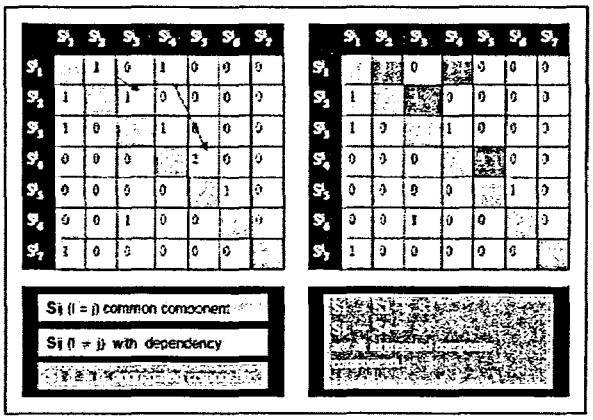


Figure 2. Parsing technique in common component

The reason we use a parsing matrix to represent the significance factor is to try and derive dimensional vectors by means of different patterns. Both $\Phi^*(S^k_i)$ and $\Phi^*(C^k_i)$ statistics are an example of CCSA parsing analysis associated with atomic service components in provisioning upward service through the alignment of S-C, S-S and B-S transformation. Figure 3 shows the positioning of common components in enterprise architecture in constructing the skeleton of the service bus for end-to-end processes.

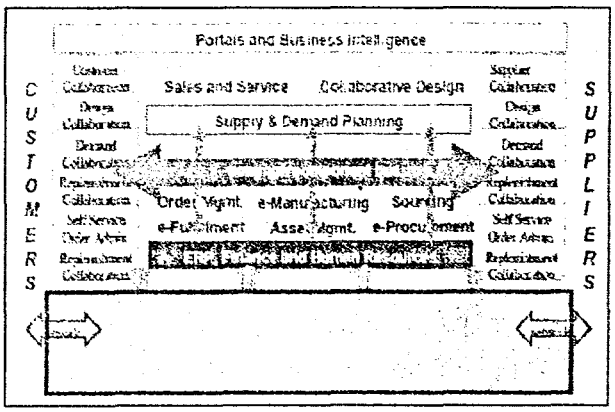


Figure 3. Common component positioning in enterprise architecture

W. R. Wu, "Determination of Common Component Integration", Association for Computing Machinery New Zealand Bulletin, 2013, [illegible] (ISSN 1176-9068)

Conclusion

This study investigates the feasibility of enterprise interoperability by applying common components as the enterprise skeleton. The importance of common components is that they eliminate grain boundary when they form the enterprise skeleton, stabilizing the component-based service and increasing the visibility of architecture alignment.

A common component can be a business component, service component, common interface, database access, security function or physical element which is frequently used in the enterprise level. Once the common component is identified, the metadata will be captured and stored in repository, and interoperability between common components will be investigated and implemented [10]. As these common components are the foundation in constructing enterprise architecture, the determination needs to be very cautious, and the parsing iteration process must be strictly executed.

Enterprise interoperability is built on the foundation of a validation process of decision, design and provisioning stages. Service component transformation means a new service is produced based on the transformation of component interference. As the business block must be deconstructed first into atomic elements in a top-down approach, and technical assets must be consolidated and aligned to service components in a bottom-up approach, both approaches may support a variety of services which come from functional or non-functional aspects. We can easily see that the common component requires a service component transformation process to achieve the objectives of enterprise interoperability.

In component-based design, componentization, analysis and validation, parsing techniques, metadata repository, vertical alignment and service provision are all key steps in producing common components. The techniques used in producing common components are mainly supported by parsing evaluation and metadata repository. Parsing techniques are an analysis process which takes the business use case and deconstructs it into atomic elements in all business, service and physical layers. The parsing techniques, coupled with metadata repository, will identify common components and aggregated components in constructing the enterprise skeleton.

This study identifies enterprise interoperability as the objective in resolving issues in enterprise misalignment and development silos. It proposes common components as a skeleton of the enterprise prebuilt foundation; it also proposes a method of validation and transformation for the further development of common components.

References

1. David Wang, (2000), "Automated Semantic Correlation between Multiple Schema for Information Exchange", MIT Sloan School of Management.

2. J. Hunter , (2001). "MetaNet - A Metadata Term Thesaurus to Enable Semantic Interoperability Between Metadata Domains", Journal of Digital Information, Volume 1 Issue 8

3. S. Dwarkadas K. Gharachorloo L. Kontothanassis D. Scales M. L. Scott R. Stets, (1998) ," Comparative Evaluation of Fine- and Coarse-Grain Software Distributed Shared Memory". Cambridge Research Laboratory

4. Traugott Koch, UKOLN, (2006), "Electronic thesis and dissertation services: semantic interoperability", E-Thesis Workshop, Amsterdam 2006-01-19/20

5. L. Chan, (2005), "Metadata Interoperability: A Study of Methodology", 3rd China-US Library Conference

6. Yu. L. (2005). "Categorization of Common Coupling in Kernel-Based Software," 43rd ACM Southeast Conference, 2005. pp 694 - 706

7. Hadzilakos Thanassis, H. G., Kavouras Marinos, Kokla Margarita, (2000). "Interoperability and Definition of a National Standard for Geospatial Data." International Journal of Applied Earth Observation and Geoinformation Vol. 2(Issue 2): 120-128.

8. Vckovski A., Brassel K. and Schek H. (1999), "Interoperating Geographic Information Systems", Second International Conference, INTEROP March 1999 Computer Science Springer

9. Curtis, K., Foster, P. W., Stentiford, F., 1999), "Metadata - The Key to Content Management Services", Proceedings of the Meta-Data '99, The third IEEE Meta-Data conference. April 6-7, 1999, Maryland, USA

E5

E5: The validation and analysis of component based metadata integration

Publication: EATIS/ACM ACM-DL May 14–17th 2007, Faro, Portugal
Euro American Conference on Telematics and Information Systems,
ISBN: 978-1-59593-598-4
Publication Contact: Prof. Amine Berqia
Chair, EATIS 2007/6/11
Univ. Algarve, Portugal
chair@eatis.org

IEEE

Date: Tue, 13 Mar 2007 20:26:01 -0600

To: rwu_academic@yahoo.com

Subject: [EATIS 2007] Your paper has been accepted!

From: "EATIS 2007" <chair@eatis.org> Add to Address Book Add Mobile A

On behalf of the EATIS 2007 - Euro American Conference on Telematics and Information Systems, I am pleased to inform you that your work tit

The Validation Process and Component Analysis of Enterprise Integratic

has been accepted. Congratulations!

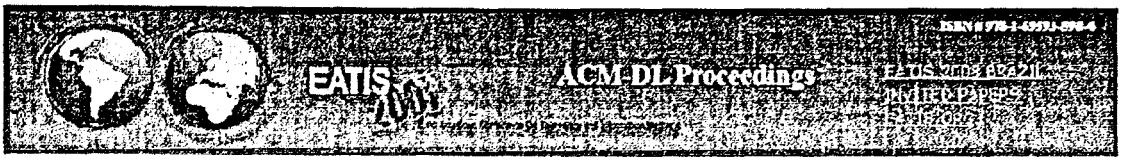
Yours sincerely,

Dr. Amine Berqia
EATIS 2007 Program Committee Chair

****

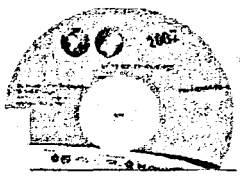
Paper is well written. The ideas are well explained and the approach i interesting. However, paper does not respect the IEEE style required b the conference.

****



EDITORIAL LETTER
KEYNOTE SPEAKERS
TUTORIALS, WORKSHOPS AND SEMINARS
EATIS 2007: FULL PAPERS
EATIS 2007: SHORT PAPERS
ORGANIZATION
CONFERENCE GOALS
MAIN TOPICS





FULL PAPERS

ID 29 - The validation and analysis of component based meadata integration
Raymond Wu, IBM
Germany

The validation and analysis of component-based metadata integration

Raymond Wu
IBM Software Group
raymond@de.ibm.com

Abstract

Component decision and validation using political and analytical models are applicable in creating the foundation of component-based enterprise architecture. This research deals with both political and analytical models to take the results from the group decision for further development. We demonstrate how to use analysis and modeling techniques to produce the common component and service aggregation which is the foundation of enterprise integration. Decision input from human factors plays a crucial role in component-based design. We have extended the focus of the methods and techniques to component decision in supporting enterprise integration and ontology interoperability, to follow the nature of business performance. As we believe human interactivity in this early analysis and pre-design stage plays a key role in the integration's success, we have assessed the feasibility and received the consensus of the pre-design stage of industry.

Key Words: Enterprise, integration, component, metadata

Introduction

We classify the steps in enterprise integration into three stages: The first stage covers the decision and validation, the second stage covers the analysis and modeling, and the third stage covers design and service virtualization. Our research in this paper focuses on the validation and analysis work. In this early stage we start from the decision-making process of business component selection to derive the service components and physical elements; this step has been dealt with in our previous work. In the following paragraphs we will determine the candidacy for common service and service aggregation by using use case parsing and modeling techniques [1].

Both political and rational models are constructed in a component-based environment. This provides the opportunity to identify each service component for its upward alignment to business and its downward alignment to the information component, with assessment of its intensity, significance, tendency and interoperability design. Before we demonstrate how services are identified, and common components and aggregated services are produced, we describe the terminology as follows.

Literature review

The main purpose in applying group decision-making in component validation was to assess and identify the impact areas starting from business components in deriving the associated services and components for further implementation. This has been the main contribution in group decision literature as this has seldom been applied in component-based development; furthermore, we took the results from component validation in common service analysis which is the small subset of ontology and subset of business. S. Melnik (1999) proposed a universal interface which avoids common models and languages, thus the generic interoperability framework was developed to facilitate the integration of heterogeneous information systems. Within this framework individual communication protocols, data manipulation languages and data are represented in a generic manner, preserving their ontological variety. This makes it possible to build a mediation infrastructure using canonical wrappers. operational requirements of integration and layered architecture for component design [2]. Based on this common framework, we have the opportunity to build the components of both functional and non-functional elements utilizing the industry standard. The standard or common applications of the component we use in provisioning the interoperability, we term a common component. DSouza specified the standard and common component as follows: "To manage models of large scale, a coherent modeling architecture is required, with standard ways to describe shared concepts, rules, patterns, frameworks, mappings, and generators." For example, the development of different applications with object models mapped to relational tables requires a single object-to-relational mapping architectural style from a shared model as a refinement framework. If all the enterprise models maintain a separation of information, engineering and computational views which are then integrated with them in particular ways, this common structure should itself be modeled in a shared architectural style model as a package pattern [3].

In our previous research on metadata interoperability we revealed, in relation to both political and rational models, how the components are derived through the validation process and how the strategy of component implementation achieves layer interoperability. In this paper we extend our

research in component design and technical implementation for both internal and interface component factors.

Validation process

The initiative of enterprise integration strategy has three key tasks to fulfill: service validation, service transformation and service provisioning. Steps 1, 2 and 3 relate to service validation, steps 4 and 5 to service transformation and step 6 to service provisioning. Validation is mainly based on the assessment of business criteria via human decision-making with the scope of validation covering each layer of B (business), S (service) and C (component element). Once business components are selected for a project with alignment to K, the derived services from B will be assessed and validated. The semantic metadata within K-B-S-C therefore provides a means to hold their matrix alignment; for design purposes, the referential integrity and semantic interoperability require that there be a repository to hold the information. The K-B-S-C metadata integration forms a skeleton in enterprise integration with the choreographic capability of retrieving each layer's metadata for the details of each component stored in the metadata repository of B, S and C. This macro to micro hierarchical structure constructs a vertical interoperability skeleton in which the different roles work together; the architect takes care of the main route of the skeleton in metadata integration and the developer takes over the skeleton's work and extends the micro design in the transformation of components and interconnectivity of service.

Service provisioning is based on an assessment of frequency of use of a component and its dependency within a project. Before implementation, the strategic KPI will first be reviewed based on a project's requirements, followed by business component selection and validation. Once the business components have been validated, the deconstructed service components and associated component elements will be evaluated and validated, using the same validation methodology [4].

The interaction between political model and analysis model

Before we proceed to the key area of these steps, we use the architecture terminology "K-B-S-C transformation" to represent interoperability between architecture layers. We begin with enterprise strategy (K), the chief area of concern for stakeholders and business managers, to derive the components in both business (B) and service (S) layers. As each layer of K (context), B (conceptual) and S (logical) are all at the conceptual or logical level, the bottom layer which is the physical component (C) forms a loosely coupled alignment with S in provisioning bi-directional interoperability. The micro mechanism between these two layers forms the service unit; aggregate individual service provides service in the business layer to finally achieve the strategic goal:

K: KPI - strategic Key Performance Identifier and all concerns of the stakeholder

B: Business component - each block performs independent business

S: Service element - logical service component at the atomic level

C: Component element - physical component associated with S and B

We start from the political model – the decision-making process of business component selection to derive the service components and physical elements. The decision-making process uses the group decision-making method to take each individual opinion or preference and multiply it [5]. The decision makers are motivated to communicate and share opinions within the group in making business component selection. We use the same political model in all business, service and physical component validation, this is because we use the framework in keeping alignment between these different components in building up the decision and significance assessment of their interoperability, instead of merely a particular component as a result.

Both political and rational models are constructed in a component-based environment; this is because a scrambled development environment provides no opportunity for group decision-making or rational analysis. With all the contributing factors in a mixed code, any changes or add-ons can cause a huge impact. Even the scope of the impact area and components can be difficult to determine, thus the impact is high while the interoperability is low. In conventional development, weak visibility between business, service and information makes the linkage and enterprise interoperability impossible. Once the alignment is missing or invisible between layers, common services and service virtualization become impossible as these require a foundation in both macro interoperability and micro mechanism. This results in a silo system with serious limitation of services, and weakness in enterprise integration [6]. As a matter of fact, under these circumstances enterprise integration gains no benefit from group decision-making and deviates from the nature of the business. As mentioned above, the methodology we used in this research work included the component model, the political model, and macro metadata integration; we differentiate these with conventional methods according to the criteria listed in Table 1.

In the use case parsing process, first the finite service matrix S_{ij} ($i = 1, 2, 3 \ldots n$, $j = 1,2,3 \ldots n$) is plotted in both vertical and horizontal dimensions in order to predict both the common degree and the degree of sequential

dependencies. We have designated the map produced by the plotting a CCSA (common component and service aggregation) map; the process is called a CCSA process. Each service S, when plotted in the CCSA map, refers to the aggregation of service aggregation. This means that a combination of service components, if they follow a certain sequence with a certain frequency, may qualify for service aggregation; that is, two or more atomic service components can be aggregated upward to a coarse-grained service in which the grain size and threshold is determined by the architecture group using either empirical benchmarking or optimization in the evaluation of advantage (i.e., coarse grain helps in minimizing interfaces) and disadvantage (loss of flexibility in the new configuration). In the evaluation of the significance factor of service component in K-B, B-S and S-C transformation, let

$P(X^k_j)$ be the jth member's weight for kth KPI

$\Pi(X^k_{ji})$ be the jth member's preference of component X_i for kth KPI

$\Phi(X^k_i)$ be the generic significance factor which is a concatenation between the weight of the role and preference of component X_i for kth KPI

$\Phi^*(S^{kb}_i)$ be the business significance factor in the B (business) dimension, which is a concatenation between the weight of role and preference of component S_i for kth K (KPI) and bth B (business component)

$\Phi^*(C^{kb}_i)$ be the physical component significance factor in the I (information component) dimension, which is a concatenation between the weight of role and preference of component C_i for kth K (KPI) and bth B (business component)

Both $\Phi^*(S^k_i)$ and $\Phi^*(C^k_i)$ statistics are examples of CCSA parsing analysis associated with atomic service components in provisioning upward service through the alignment of S-C, S-S and B-S transformation. We introduced the structure of the transformation with focus on the S-S step. Aggregation in the service layer is important as it eliminates boundary configuration, increases the integrity of common services in an error-free configuration environment and, furthermore, can be developed into coarse-grain processes or processes adjacent to the business layer. Therefore the referential alignment of B-S strategy uses coarse grains to support business components directly; those fine grains at the bottom of the service layer can form a different pattern in supporting the coarse grains. This implies that the fine grains serving as the foundation of the services can be reused so that one service can contribute to provisioning many coarse grains through different patterns. The formation of a pattern can use common components as a base, as they are common in provisioning generic services in each business component. Aggregation that is based on a pattern derived from parsing analysis as a usual case is crucial in common services; in the terminology our use of "common" indicates an encapsulated enterprise service that is seldom changed even though a light revision is acceptable.

As aggregated services and common components are critical in forming the enterprise skeleton, the design and aggregation process must be cautious from an interconnectivity perspective. This implies that high integrity, optimization, grain size, dimension in service provisioning, visibility, traceability, performance and other factors are areas which must be brought into consideration.

Modeling and validation techniques

In the following paragraph, we demonstrate how to use the identified steps to derive the coherence of their interference in building up enterprise integration. Before we proceed to the key area of six steps, we use the architecture terminology "K-B-S-C transformation" to represent interoperability between architecture layers. We began with enterprise strategy (K), the chief area of concern for stakeholders and business managers, to derive the components in both business (B) and services (S) layers. As each layer of K (context), B (conceptual) and S (logical) are all at the conceptual or logical level, the bottom layer which is the physical component (C) forms a loosely coupled alignment with S in provisioning bi-directional interoperability. The micro mechanism between these two layers forms the service unit: aggregate individual service provides service in the business layer to finally achieve the strategic goal.

Common components through parsing analysis

Once the validation of business components and their associated service components is completed in steps 1, 2 and 3, we can move on to service provisioning based on the components we selected. The provisioning is an architectural process to produce both common and composite components by means of use case parsing to count the frequency and dependency of service components. Frequency means a count of the appearance of a component in the parsing process; dependency means that two or more components work together frequently and follow a particular sequence in forming a composite service or process. The mapping between each layer is shown in Table 1.

In the use case parsing process, first the finite service matrix S_{ij} (i = 1, 2, 3 ... n, j = 1,2,3 ... n) is plotted in both vertical and horizontal dimensions in order to predict both the common degree and the degree of sequential dependencies. We have designated the map produced by the plotting a CCSA map; the process is called a CCSA

UID	Use Case	EID	SID	Service	CID	Component
	Buyer sends payment information to the merchant server.			Virtual (payment_info)		Buyer_req_merchant_payment (info)
				Virtual (payment_security)		Merchant_exec_buyer_payment (security)
				Online (auth_info)		Merchant_exec_buyer_messging
	Merchant software takes payment information from the cardholder and sends it to the acquirer (merchant's bank).			Virtual (payment_info)		Merchant_submit_acquirer_payment
				Online (auth_tx)		Acquirer_authenticate_merchant_payment
				Security (merchant)		Acquirer_validate_merchant_tx
				Virtual (payment_merchant)		Merchant_charge_acquirer_computing
	Acquirer sends an authorization request to the issuer over the interbank network.			Cardholder (status)		Acquirer_req_issuer_auth
				Account (3rd party)		Interbank_reps_acquirer_auth
				Online (auth_interbank)		Issuer_respond_interbank_auth
				Security (cardholder)		Interbank_check_cardholder_fraud
	Acquirer notifies merchant about the status of authorization			Cardholder (status)		Acquirer_notify_merchant_status
				Account (merchant)		Merchant_check_acquirer_status
				Recur (merchant)		Merchant_fulfill_acquirer_order
	Merchant presents the charge to the acquirer bank.			Security (merchant)		Merchant_charge_acquirer_verify
				Virtual (payment_merchant)		Merchant_charge_acquirer_computing
	Acquirer sends a settlement request to the issuer.			Settlement (batch_run)		Acquirer_run_issuer_settlement
				Settlement (batch_report)		Acquirer_send_issuer_settlement
	Issuer charges the buyer's credit card account and at regular intervals notifies buyer of the transactions			Virtual (payment_issuer)		Issuer_charge_by_interbank
				Cardholder (status)		Issuer_run_report_buyer (accumulated)
	Buyer pays the charges to the bank. Acquirer credits merchant's account			Account (buyer)		Buyer_pay_issuer_interbank (credit card)
				Account (acquirer)		Issuer_credit_acquirer_buyer
				Account (merchant)		Acquirer_pay_merchant_issuer (buyer)

Table 1. Component mapping between each layer

process. Each service S, when plotted in the CCSA map, refers to the aggregation of service aggregation. This means that a combination of service components, if they follow a certain sequence with a certain frequency, may qualify for service aggregation; that is, two or more than two atomic service components can be aggregated upward to a coarse-grained service in which the grain size and threshold is determined by the architecture group using either empirical benchmarking or optimization in the evaluation of advantage (i.e., coarse grain helps in minimizing interfaces) and disadvantage (loss of flexibility in the new configuration).

Service aggregation through parsing analysis

Both $\Phi^*(S^k_i)$ and $\Phi^*(C^k_i)$ statistics are shown in Figure 1, as an example of CCSA parsing analysis associated with atomic service components in provisioning upward service through the alignment of S-C, S-S and B-S transformation. Aggregation in the service layer is important as it eliminates boundary configuration, increases the integrity of common services in an error-free configuration environment and, furthermore, can be developed into coarse-grain processes or processes adjacent to the business layer. Therefore the referential alignment of B-S strategy uses coarse grains to support business components directly; those fine grains at the bottom of the service layer can form a different pattern in supporting the coarse grains. This implies that the fine grains serving as the foundation of the services can be reused so that one service can contribute to provisioning many coarse grains through different patterns. The formation of a pattern can use common components as

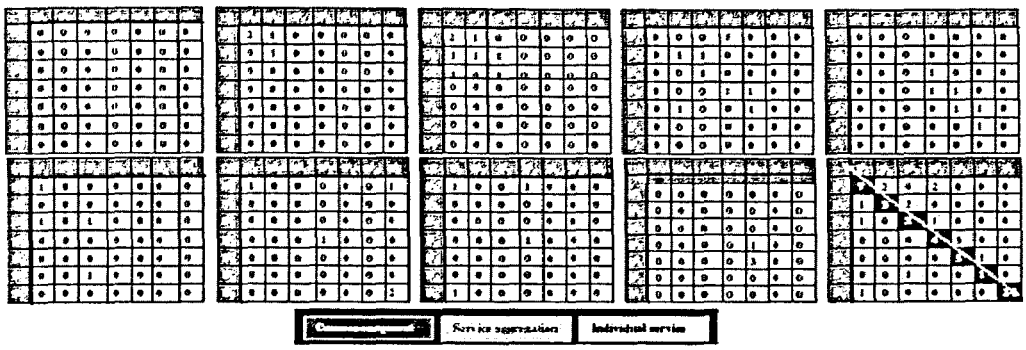
The information aggregation in the component layer forms the metadata that is stored in an information repository distinct from the service components. It constructs part of the component element in the physical layer. The component element consists of physical units of functional and non-functional elements; while an agent is an example of a functional element, a business object or a security component are both considered non-functional elements.

Service aggregation is a process for deriving the metadata within a layer, so even though the service components and component elements have been well defined in Steps 1 through 5, fine tuning is required to derive the final step.

In Step 4 we described how the service component is aggregated. Within the component layer there is no such process: as the physical layer is at an atomic level loosely coupled with peer components and associated services, we assume all the component elements in information component form are in an optimized state and that no further aggregation is required. Therefore, before any information component is defined, a suitable aggregation in optimization is necessary; this implies that the aligned service component is always associated with an optimized component. Thus, if the component is an information component, there is no further aggregation required after S-C is configured.

Determining implementation priority within enterprise integration

In each K implementation, the interoperability is built by using semantic business deconstruction to derive service



Figures 1. Parsing techniques in producing common component and service aggregation

elements and component elements; at this point each element can be aggregated and composed into composite components or identified as a common service for reuse and shared service purposes. There are 3 years of assessment in determining the tendency of each service component.

- Business layer - Group decision to determine which business component to use
- Service layer - Group decision with some analysis work from architecture
- Component layer - Group decision with some analysis work; an information pre-built foundation in meta form is required to make sure the information metadata is well integrated in this layer
- Service transformation - Identify services in relation to common and aggregation

Conclusion

In role-based group decision-making, each role has a different weight in each layer within B-S-C, and each group member has a different preference in their component selection. We use the political model, starting with the business components. We repeat the political model in a use case deconstruction to derive service components and physical elements. The group members validate both service components and component elements by using the same method with differently weighted roles. The evaluation between service components and physical components is based on how significant their contribution to the business goal is from a logical and physical information component perspective; a higher degree of information aggregation of C within B-S-C gives a higher preference weight in that physical component.

The multiplication of B and S significant factors represents the significance of S within the K-B-S-C transformation; using the same method, the multiplication of B and C significant factors represents the significance of C within the transformation. The objective of K-B-S-C transformation and significance factor evaluation is to try to build up the vertical enterprise skeleton and extend the key services to their peripheral surroundings in performing full service virtualization.

All the service components need to be deconstructed into atomic elements as both top-down or bottom-up approaches encounter of a variety of services which may be functional or non-functional. Unless we understand the specifications of each element, we should not make the assumption that deconstruction can be stopped in the middle of the process and that a coarse-grained service has sufficient support for a particular business function or use case. This implies that a full deconstruction and aggregation are necessary for all service components, and the analysis model supports deconstruction and aggregation as complementary processes of the political model in achieving the early stage of enterprise interoperability.

References

1. Remko Scha, Rens Bod[5] and Khalil Sima'an, (1999), "A Memory-Based Model of Syntactic Analysis: Data-Oriented Parsing" Special Issue on Memory-Based, Language Processing) 11. pp. 409-440.
2. Sergey Melnik, (1999), "Generic Interoperability Framework", Distributed and Parallel Databases Journal 1999
3. L. Chan, (2005), "Metadata Interoperability: A Study of Methodology", 3rd China-US Library Conference
4. Guangquan Zhang and Jie Lu, (2005), "The Definition of Optimal Solution and an Extended Kuhn-Tucker Approach for Fuzzy Linear Bilevel Programming", IEEE Intelligent Informatics Bulletin November 2005 Vol.6 No.2
5. Jie Lu, Guangquan Zhang & Fengjie Wu, (2005), "Web-based Multi-Criteria Group Decision Support System with Linguistic Term Processing Function", IEEE Intelligent Informatics Bulletin June 2005 Vol.5 No.1
6. Raymond Wu, (2006), "Enterprise Vertical Integration in the e-Government", Electronic Government, an International Journal (EG) ISSN (Online): 1740-7508 - ISSN (Print): 1740-7494
7. DSouza,(2001)," Model-Driven Architecture and Integration Opportunities and Challenges", Kinetium
8. M. Staab, N. Stojanovic, R. Studer, and Y. Sure, (2001), "Semantic Portal - The SEAL approach. to appear in: Creating the Semantic Web.", MIT Press
9. O. Valkeapää and E. Hyvönen, (2006), "Semantic Annotation with Browser-based Annotation Tool", 1st Asian Semantic Web Conference

E6

SOA Maturity Model evaluation

IBM

Version Number: 1.0
Date: 10/10/2006
Author: Raymond Wu, Consulting IT Architect
IBM Software Group

For the presentation to ASEAN in Financial Industry enterprise architecture

Background

Business service driven technology play crucial role in new IT industry, the key definition points for a business service are that it must:

1. Represent a function
2. Be offered at a granular, but functional, level
3. Be offered at a larger-grain level as an aggregate of fine grain services
4. Permit integration via multiple technologies directly to the service (web services) for both loose and tight coupling of the service, as opposed to forcing all integration through message queuing
5. Be able to be combined with other services into composite transactions
6. It must permit customization of business logic within the service

The coarse grain element is the aggregation of fine grained element and to decompose business into atomic level element, we understand the following atomic level element and composition:

Business level - common or individual

Common module:	A business common task unit, no need to breakdown
Interface:	A calling point to other process/terminator in finite state
Comm/Ind. task:	A common or individual task, no need to breakdown
Report & Query:	Common or individual report or query screen in finite state
Registry:	Meta definition for virtual layer, logic and flow

System level - common or individual

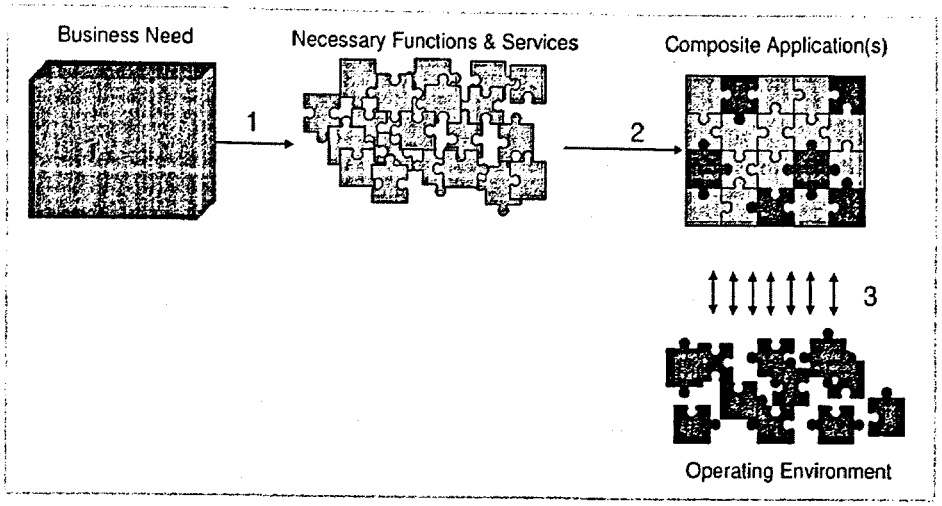
Common utility:	A common system utility unit, no need to breakdown
Common screen:	A common UI screen in finite state
Comm/Ind. event:	A common or individual system level task, no need to breakdown
Repository:	Meta Definition of Code, DB, logic and Parameter

Following is an example from our local consulting and development for how to use these components to provision a business function;

The following example is a macro scope of how each coarse grain element interact each other and work for a particular business component

Decomposing business needs into necessary services and composing combinations of existing and newly created services that represent business processes.

1. Decompose the Business Need into its necessary function and services (Separation of Concerns)
2. Reuse/Create Business Application Specific function and services (Create/reuse services)
3. Utilize common services provided by the Operating Environment (Leverage Infrastructure)



Service Integration Maturity Model (SIMM) - the framework for assessment, transformation and roadmap creation, consists of following three steps:

Transformation
A set of organizational changes of state to support business imperatives and goal

Adoption
How an organization absorbs and assimilates change locally and globally as it "digests" or gets acclimated to the changes.

Maturity
The characteristics and behavior of parts of the organization that are undergoing transformation and are undergoing adoption

The objective of SIMM is listed as followed:

- Assess client's current state in service integration and flexibility (including services orientation) and their desired or future state, for a line of business or enterprise
- Provide a model for assisting a client in determining its architectural strategy when adopting service orientation
- Improving pain points in areas of flexibility or integration
- Provides an architectural roadmap for one or more initiatives in legacy transformation, package implementation or integration, application renovation or systems integration
- Provides a model for determining scope, focus and incremental steps (i.e., architectural roadmap) for a transformation towards service orientation with increasing integration maturity with defined business benefits)
- Provides a framework for surfacing insights and identifying IT improvements in areas of component development, SOA, services integration, and IT processes (e.g., governance) improvements

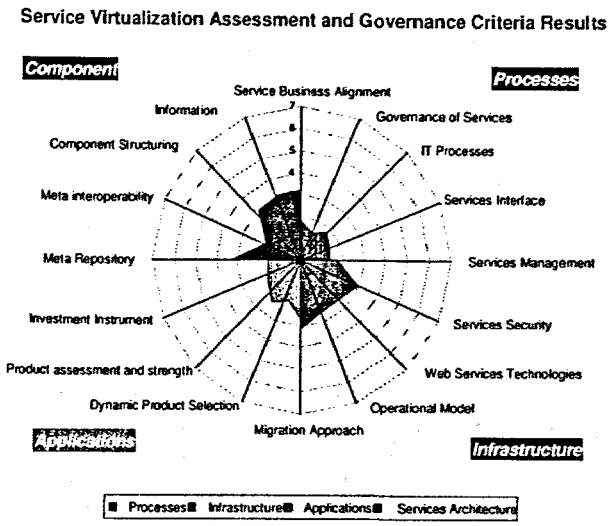
SIMM facilitates the adoption of on demand for an enterprise with its focus on increasing levels of flexibility for five dimensions

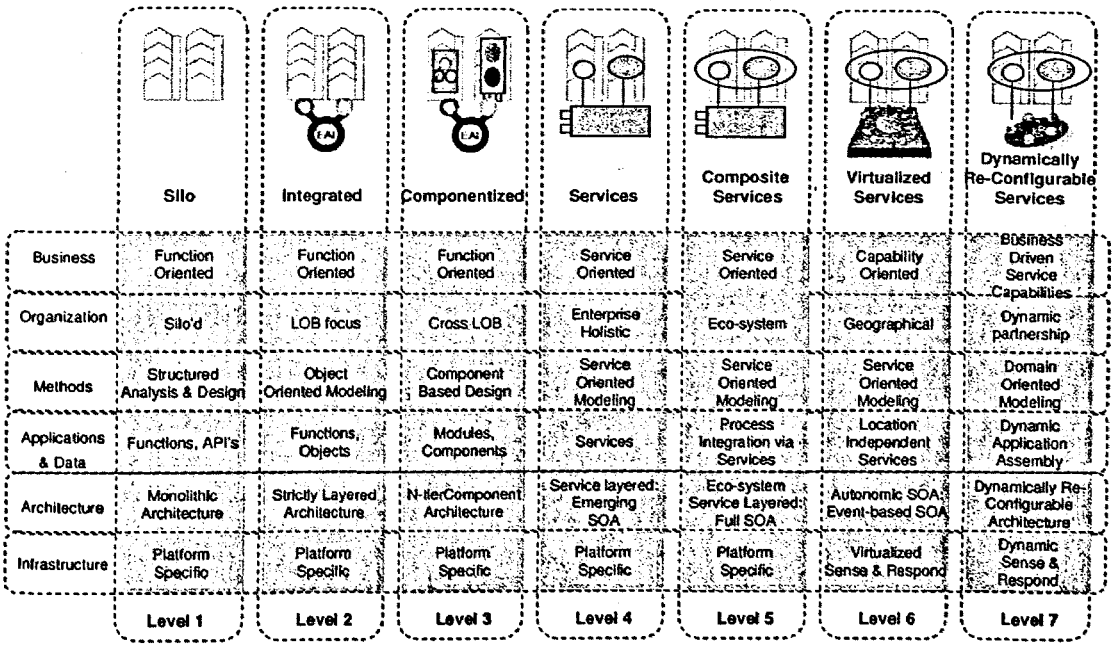
- Business View of IT, methods and processes, application portfolio, architecture, infrastructure
- The above aspects of an enterprise or line of business achieves flexibility for on-demand when integration is planned and not an afterthought;
- Application software is constructed and designed, *mindful of flexibility.*
- Increasing flexibility is the key, for business models, processes, applications and the infrastructure.
- Technology must act as an enabler not a barrier. This requires that integration be planned as part of building or transforming an application portfolio

SIMM Maturity

SIMM is used to conduct assessments and determine level of maturity for an enterprise or line of business and design a plan of action

- SIMM is used to 1) determine current maturity level of integration, 2) determine which characteristics are desirable to achieve and 3) defines how to achieve them by attaining a new level of maturity.
- This will determine whether problems encountered at a given level of maturity can be solved by evolving to a higher level of integration.





Service Oriented Architecture and Component Based Implementation

Components without SOA

Potential benefits

- Software solutions developed rapidly from existing components
- Economies of scale achieved from reusing available components
- Choice of different components yields some pre-implementation flexibility

Potential drawbacks

- Components are permanently assembled into applications – easier to "plug" than to "unplug"
- Lack of post-implementation flexibility

SOA without Components

Potential benefits

- Services developed rapidly from existing applications and packages
- Opportunities for reuse and economies of scale
- Useful path towards further reengineering

Potential drawbacks

- Inflexibility behind the service interface.
- Existing applications and packages preserve their legacy limitations
- Concerns about scalability, portability of implementation,
- Reengineering may never happen

Components and SOA

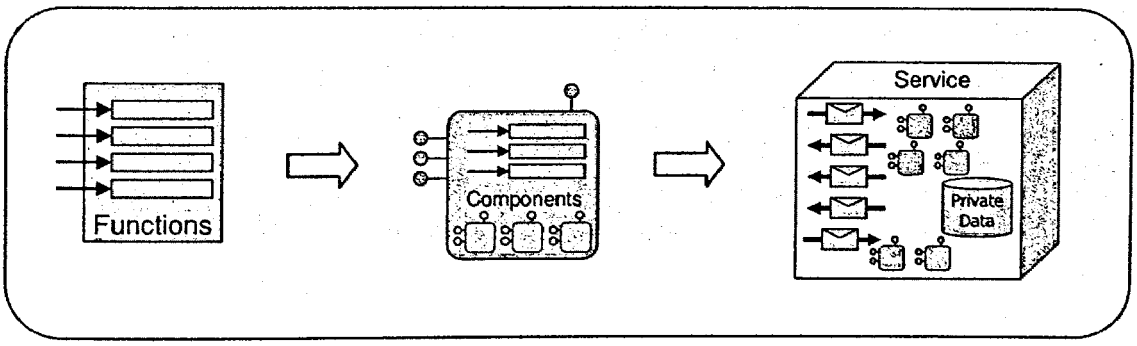
The true potential of components and SOA only really comes when you have both of them.

- Components and services assembled with loose coupling, and dynamic recoupling.
- Coherent service interfaces supported by flexible component-based applications
- Service-based approach applies both inside and outside the application
- Scalable, flexible, on-demand solutions

Component Transformation

There has been a natural evolution: Function -> Component -> Service and Services comprise messages, components, and data

- ✧ The private data is completely encapsulated by the service
- ✧ Messages are the only ways into and out of the service



Techniques



T1 Semantic Mediation in Micro Component Design of Enterprise Integration
IJMSO

T2 An Industry Case Study of MicroComponent Design & Semantic Mediation
IJMSO

T3 Micro Process Design in Component based Architecture
Haamaha

T4 Implementation Priority and Dependency in Component based Design
IEEE-EISWT

T5 Micro Process Design and Services Virtualization
IEEE-Southeast

T6 Metadata Computing in CRM
Oracle Corporation

T7 Event Driven and Batch Process
Oracle Corporation

T8 Database Partition and CLOB
Oracle Corporation

T1

T1: Semantic Mediation Modeling in the Service Component Design

Publication: IJMSO Journal 2007

International Journal of Metadata, Semantics and Ontologies (IJMSO)
(http://www.inderscience.com/browse/index.php?journalCODE=ijmso)

Publication Contact: Prof. Miguel-Angel Sicilia
Editor in chief, IJMSO Inderscience
Computer Scince, Technical School of Informatics
University of Alcal, Spain
msicilia@uah.es



From:	"Miguel-Angel Sicilia" <msicilia@uah.es> Add to Address Book Add Mobile Alert
To:	"'Raymond Wu'" <rwu_academic@yahoo.com>
Subject:	RE: RE: Submission for your review for publication purpose - IJMSO
Date:	Thu, 31 May 2007 13:03:58 +0200

Dear Raymond,

The papers are accepted and scheduled for publication, so only minor c
formatting changes will be required as they go through the publishing
process. You can safely mention them as definitely accepted in the
record of
your Ph.D. dissertation, even though the final publication date is
still
subject to uncertainty.

* Semantic Mediation Modeling in the Service Component
Design

* An industry case study of micro component design and
semantic mediation

Best regards,
Miguel-Angel

International Journal of Metadata, Semantics and Ontologies ISSN: 1744-2621

Semantic Mediation Modeling in the Service Component Design

Raymond Wu

IBM Software Group
Email: rwu_academic@yahoo.com

Abstract: *Service transformation of semantic mediation is one of the big challenges in this century from the architecture scale and computing complexity. The main objective of the mediation is to increase IT service levels and minimizing architecture gap, consequently enhance service base productivity and reduce computing expenses. Achieving this objective requires multi-dimensional approaches including business innovation, service strategy, solution architecture, implementation and integration. Therefore, architecture alignment and roadmap coherence are extremely important. We further extend our investigation of semantic mediation by using AHP pair-wise comparison and eigenvalue techniques in component evaluation. Consequently, any new additions or changes in the business frontier can cause significant impact. This has led to architecture risking "re-design" instead of "re-use."*

Our proposal mainly argues that first the terminology of semantic mediation needs to be clarified. This should not be initiated from the macro architecture layer; instead we propose that both mediation and interoperability should impact components first and push semantics strategy directly into the micro process to be synthesized into the service layer.

Keywords: semantic mediation, enterprise integration, service component

Biographical notes

Raymond Wu is the IT architect currently working with IBM Software Group, his expertises include, enterprise architecture, integration, metadata, service transformation, financial, telecom and database design. Previously he worked with Oracle, Citibank and Vodafone as enterprise architect.

Introduction

The transformation itself is crucial as the service concept is changing rapidly. The coherence and roadmap are also critical as most of the service providers and vendors are still lacking in enterprise vision in consolidating a silo-approach into a global common practice. This implies a misalignment between direction (value and context), strategy (innovation, solution and architecture) and execution (implementation, integration and roadmap) and results in a weak driving force toward global compliance. Thus, the service terminology and concepts relating to different aspects (i.e. customer-centric, master service data, single customer view, service insight, portal service, service channel consolidation, call center agent and other factors) do not have enough visibility for one to see how they relate to and impact on each other. Evidence of this is found in the variety of architecture views as well as dispersed integration definitions and terminology. These factors combine to create a bottleneck in global common practice (Gruninger M., 2005). Our integration approaches are based on the metadata component which is built on top of the foundation of semantic interoperability. This means our method is not based on peer-to-peer data or system or application integration, rather our interest is in following the nature of business. This interest includes how the business is composed and what strategy synthesizes the

Copyright © 2007 Inderscience Enterprises Ltd.

semantics is automatically inherited from metadata when data is abstracted in meta form, semantic mediator need be implemented within service based ontology and this study like to propose a multi-disciplines and multi-dimensions ontology, to produce service dimension and its pattern. This study also use experiments to benchmark each pattern in determining semantic strategy, as different process and strategy imply different pattern. Our semantics strategy is focused on architecture interoperability by mediating components in different form. Thus the nature of semantics itself and mediating techniques are all our interests. There are two areas of semantics need to be clarified: the first one is business semantics which is strategic semantics or what the business wants to do. The second area is semantics content i.e. the information needed in responding to the service request to achieve a particular business semantic (Moulton A., 2003). Gugliotta A. (2006) argued that, dealing with interoperability among heterogeneous knowledge sources, following mediation requirements and solutions need be considered: i) Data mediation: At a semantic level, different concepts can be mapped through mediators. ii) Goal mediation: Multiple Web services can be linked to the same goal via mediators. iii) Process mediation, Each Web service presents a choreography describing how a client talks to the deployed service. Our focus as regards interoperability is how to mediate these two areas of semantics in achieving a business goal by a particular dimensional service. Apart from that, event trigger technique is also an important area in achieving dynamic semantic mediation.

Service-Oriented Architecture (SOA) is commonly launched for Enterprise Integration and service base implementation. Semantic annotation of all aspects of Web service, including their capabilities, data and processes, is an important step towards making SOA a success. It was proposed by Pedrinaci, C. (2006), the Event-Driven Architecture (EDA) Coupling with SOA to obtain a particularly appealing architecture for supporting the effective execution of possibly distributed and completely decoupled services, synchronously or asynchronously executed in an efficient manner. Event-based communication is indeed a useful way for a decoupled integration but it does not, and it actually cannot, avoid the need to agree on the events and the need for mediating between heterogeneous data models . Designing new systems will require establishing an agreement on which events there need to be, their syntax and their actual semantics. Thus, adapting or, as we prefer to call it, mediating between data models remains a critical requirement.

Apart from the similarity been found in above between our concept and existing literature, we like to outline the differentiators of our approaches from the existing common approaches:

1) From semantics perspective, our semantics and mediation approaches differentiate itself from conventional concept due to following reasons: The semantics concept we introduced consists of three level derivation: i) technical level in atomic element form, ii) syntax level about elements' interconnectivity and iii) context level which is environment dependent. None of these three levels can be omitted as the alignment within these three levels is the key to take human decision and vision into technical realization. The coherence of three level semantics has not been found from our literature review as most of the related literature only focus in one or at most, two levels from above, therefore the service interoperability we proposed is also differentiate itself from existing literatures as most of the literature limited their approaches of semantic interoperability to the macro integration of system or application level, and as our approach is based on micro and macro process synthesis, so each atomic element has alignment with business, service and technical layer in achieving a vertical interoperability, meanwhile, the choreography of atomic element gives opportunity of fine grained components to form coarse grained service and common service in achieving macro service interoperability within enterprise, and this is the fundamental differentiators compared with conventional concept.

Copyright © 2007 Inderscience Enterprises Ltd.

2) From mediating perspective, we define our semantic mediation as the means of semantics transmission through multi-dimensional approaches. First, dimension is component validation between human and machine through decomposition and aggregation processes. Second, dimension is semantic transmission through metadata design and synthesis of micro and macro processes. The third, dimension is semantic implementation under enterprise vision, so once the three levels of semantic mediation is implemented, it gives opportunity for atomic components be designed in a synthesis way which consists of micro and macro processes. Meanwhile, as the visibility of vertical alignment is clear between architecture layers, the semantic centric metadata forms a clear picture of the enterprise skeleton stretched across the interoperability between business, service and information. Through a semantic mediation approach, we can either reuse the metadata mapping and configuration from that successful experience or construct a new map for the new requirements by using the same methodology. Consequently, once the service components and aggregation services are built they can either be reused or aggregated to a coarser grain if the aggregation is common and optimized in the parsing evaluation.

3) From industry standard perspective, our approach of semantic mediation has been expected to produce significant benefits to the industry typically in the following scenarios; i) large scale global-local topology, that the enterprise architecture require a semantic metadata centric solution to maintain the coherence of semantics across organizations in synchronization with the context of corporative strategy, ii) the interactive between business and IT is significant, that company's KPI (key performance index), service and infrastructure requires effective semantics to transmit across vertical direction and ontology of the alignment require identification of common components and method of rapid implementation for those new alignment of semantics, and ii) service virtualization in industry's main route become a high demand, for instance, eTom in telecom industry and Zachmann enterprise framework works widely as industry foundation however, it has been insufficient to many scenarios such as join-venture, acquire and merge cases and the main reasons are industry main route is still far from common practices due to the industry semantics standards are still in macro process level instead of micro element, as a matter of fact, many integrations are still in case-by-case or n:n situation and lack in a robust driving force works as a single source of truth in both method and semantics to drive the transformation into a n:1 evolution, we consider the root issue came from the weakness in semantics synthesis process, in the other word, the mediation of services has not been enriched. Given the approaches we proposed, the application of our concept has already been implemented in several projects with convincible advantages from their deployment.

Notation of Semantic Mediation and Service Interoperability

We subsequently performed three emulation experiments on semantic mediation to aid in our understanding of the behaviors of semantic metadata, service mediation and integration. The experiment was conducted in the separate research to use semantic computing to emulate semantic mediation. In the other words, we use computing model to emulate service model.

Experiment I - In our first experiment, we use an SQL-based computing model to emulate the phenomenon of semantic interoperability. The computing model consists of the parser, the optimizer, and the row source generator which form the SQL compiler. Compiling the SQL statements into a shared cursor, and associated with the shared cursor, is the execution plan.

Experiment II - In our second experiment, we used a similar model to experiment I to emulate the scenarios of random query and semantic cursor computing, and produced an execution plan as well as a physical read and response time.

Copyright © 2007 Inderscience Enterprises Ltd.

Experiment III - For the scenario we used in experiment III, the source site is the Unix customer management system and the remote sites are the customer care center (MS SQL-server case) and call center (Oracle/Unix case); both cases can be expanded to an unlimited or heterogeneous database, a different operating system and different data format.

From our observation and project experiences, our motivation for the development of semantic technologies did not originate from a conventional "divide-and-conquer" mindset, which merely takes the output from business analysis and interpretation for further development. In our literature review regarding this research field, we saw that some crucial points were missing. We consider this crucial because within enterprise, the skeleton of dimensional service with linkage between business, service, component and information is mandatory. By means of these dimensional services, enterprise and industry have the opportunity to provision their strategic dimensions with further impact on patterns and common components. The assumption is that industry requires a certain level of maturity in common practices with regard to service dimensions and the derivation process in achieving realization. A typical example is the eTOM level 3 in the telecom industry. Our priority in service transformation is the knowledge of how to create the visibility of the alignment within these dimensions and how the components, either in conceptual, logical or physical forms, mediate each other with linkage to the skeleton of service architecture. We use the following terminology to represent semantic notation:

Service dimension: A group of service components and alignment in provisioning a particular business objective; examples include customer insight and pre-paid services. A service dimension is different from a use case, as use case is in daily operational form which is a virtual process directly decomposed from the business block, while the service dimension is a strategy for achieving a strategic business feature.

- ✧ Service pattern: The pattern used in provisioning a particular service dimension. Pattern is a symbolic combination of components used by following a particular rule or principle. We wish to demonstrate how the service dimension is represented by a mathematical model.
- ✧ Semantic mediation: The virtual semantic service in mediating components or contents in provisioning combinational services.
- ✧ Service interoperability: The strategy and realization of architectural interoperability by using semantic mediation strategy in provisioning the enterprise level service.

From the results and assumptions gained from the experiments, we are aware that our motivation is in extending our knowledge of the cache memory mechanism to service mediation. We will now turn to the investigation of the mediation of semantic transformation within architecture in a vertical direction. First, in service Cartesian space we use the following symbols in representing component-based alignment and ontology:

'Φ^n' denotes n's service dimension of Φ

'S^n_m' denotes m's service component of Φ^n

'C^n_{mk}' denotes k's component which support S^n_m

'Π' denotes the semantic interoperability of representing the mediation between Φ, S and C.

From the service perspective, Π is an ontological representation of the interaction space of the service patterns which is in the enterprise level while Φ^n, S^n_m and C^n_{mk} are all in the service dimension form. In service Cartesian space we use the following symbols in representing semantic mediation and interoperability:

'$\bigcup$' denotes the union of macro integration without decomposition

'Σ' denotes the summation from an individual component without mediation

Copyright © 2007 Inderscience Enterprises Ltd.

'×' denotes concatenation of semantic mediation in component-based integration

It is easy to understand that '∪' means the binding glue of a large block or architecture layer; 'Σ' means creating components and using them in an ad hoc base without effective mediation. '×' means semantic mediation and aggregation are all in a mature stage, thus we can reuse the pattern and build a common foundation; for each service component or dimension we have no need to start from scratch. Second, we classify service patterns into several different groups and investigate the trend and micro process for pattern benchmarking; in this way we can observe their behavior by using their leveraged computing model.

Service Pattern Classification

As mentioned above, a service pattern is the implementation and realization of dimensional design that consists of the principles and methods for how components interact with each other; they can be classified into the following four patterns:

Pattern 1. Coarse-grained "glued" type: In a conventional peer-to-peer integration, service mediation is only limited to the block level; most of the components are large block and are "glued together." The mediation is thus limited to a centric repository for common access or interfacing between architecture layers such as API, data exchange, ftp, or middleware; there is no sharing or expectation of re-use. Indeed, each small change can require huge effort and be a high risk in analysis. The mathematical model for this pattern is expressed as follows:

$$\Pi = (\Sigma C_{mi}) \cup (\Sigma C_{mj}) \cup (\Sigma C_{mk}) \qquad (1 < m < M, 1 < i < I, 1 < j < J, 1 < k < K)$$

Where ΣC_{mi} represents a technical asset or large block (business or technical program) within a system, application or database which is derived directly from business without any decomposition.

Pattern 2. Fine-grained "glued" type: In a non-semantic component-based integration, we produce fine-grained interoperability between components in the construction of ad hoc type mediation; in this way the whole enterprise of different systems and applications can build their interoperability in a horizontal direction without a clear picture of the vertical direction. The mathematical model for this pattern is expressed as follows:

$$\Pi = \Sigma\Phi^n = \Sigma (C^n_{mi} \times C^n_{mj} \times C^n_{mk} \ldots) \qquad (1 < m < M, 1 < i < I, 1 < j < J, 1 < k < K)$$

A lack of visibility in the service alignment and weak enterprise interoperability results in limited reuse and immaturity in provisioning any service pattern.

Pattern 3. Thin layer service type: In a partial semantic mediation integration, which we call the thin layer service, it is possible to construct the visibility of a virtual service across enterprise; in this way part of the component has the opportunity and can start to form upward aggregation into the service layer, which is the virtual concatenation output from component. We use the following mathematical model to express the ontology of the thin layer service:

$$\Pi = \Sigma\Phi^n = \Sigma (C^n_{mi} \times C^n_{mj} \times C^n_{mk} \ldots) \cup \Sigma S^n_m \quad (1 < m < M, 1 < i < I, 1 < j < J, 1 < k < K)$$

As part of the components start to form the upward service layer, some identified business components concurrently start to explore downward decomposition to the associated components. A lack of service mediation and poor knowledge-based metadata are at this point far from mature enough to direct an effective ontological strategy though each ad hoc implementation or to aid in the tuning of mediation and common practice in semantic interoperability.

Pattern 4. Thick layer service type: In this full semantic mediation integration, the thick

Copyright © 2007 Inderscience Enterprises Ltd.

layer of service provides high agility and virtualization of the services layer in constructing a dynamic and enriched service. We can divide the thick layer service into three subgroups as follows:

4.1 Thick service layer fine-grain only:

$$\Pi = \Sigma\Phi^n = \Sigma\,(C^n_{mn} \times S^n_m) \qquad (1 < m < M)$$

4.2 Thick service layer mixed fine and coarse grain:

$$\Pi = \Sigma\Phi^n = \Sigma\,(C^n_{mn} \times (S^n_i \times S^n_j)) \qquad (1 < j < i < m < M)$$

4.3 Thick service layer transparent:

$$\Pi = \Sigma\Phi^n = \Sigma\,S^n_m \qquad (1 < m < M)$$

Pattern Benchmarking for Optimized Semantic Mediation

In the non-component base of pattern 1, the service dimension is not applied as most of the linkage between business and technology are tied up; physical binding causes black box in all business change. We call a new business feature or function either "adds-on" or "enhancements." The symbols which relate to the black box are listed as follow:

$\Phi(P^n)$: to represent business scope of n_{th} program

P^n: to represent the non-component based technical implementation of n_{th} program

$\Delta\Phi(P^n)$: to represent business scope deviation between $\Phi(P^n)$ and $\Phi(P^{n-1})$

$\Phi(P^n) => P^n$: to denote the impact projected from scope $\Phi(P^n)$ to program P^n

We can use $\Delta\Phi(P^n) = \Phi(P^n) - \Phi(P^{n-1})$ to represent the scope change in an accumulative basis; unfortunately, we cannot use $\Delta P^n = P^n - P^{n-1}$ to represent the effort spent on a program change ΔP^n associated with business change $\Delta\Phi(P^n)$, as the alignment between business and technology doe not exist or is unclear and assuming we have two separate implementation, the first one uses $\Phi(P^{n-1})$ to derive P^{n-1} and the second one needs to use $\Phi(P^n)$ to derive and rebuild P^n as we are unable to apply "enhancement" or "adds-on" to merely "enhance" or "add" that deviated scope to the existing program; apparently the ΔP, associated with $\Delta\Phi(P^n)$, impacts the whole scope as $\Delta P >> P^n - P^{n-1}$.

If we assume within the initial scope, U^i is the subset which should be frozen. This could include core technology, master data schema and system level specification. The fixed scope 'U^i' is actually the base for $\Phi(P^i_0)$ which is the initial business scope of program P^i_0 and 'V^{i+}' denotes the external work (i.e. add-on) which is not related to U^i. 'X' denotes the transformation coefficient in mediating business and technology. The larger the value of X, the higher mediation gap we will suffer. If we set X = 1, this means the scope change (either adds-on or enhancement) cause the same impact as when we build the frozen part. Moreover, if the frozen part needs to be large, this means the complexity is higher, so each change can cause a higher impact. When X = 2, which is a high value of the transformation coefficient, this usually means the "scrambled" type transformation with subsequent higher risk; each change can cause twice the impact as those frozen parts caused. In a well-aligned service environment, X may approach 0 as a pre-built component foundation already constructed; this is the base requirement and can easily cope with a change in scope.

In pattern 1, only $\Phi(P^n)$ is an incremental basis as the business scope changes; this is paperwork which can be manipulated in a linear Cartesian space and expressed by following the equations which apply to the gap between P^1 and $\{P^2, P^3 \ldots, P^n\}$

$$\Phi(P^i_0) = U^i X_0$$

Copyright © 2007 Inderscience Enterprises Ltd.

$\Phi(P^i_1) = U^i X'_1 + V^{i'}_1$
$\Phi(P^i_n) = U^i X'_n + V^{i'}_n$
$\Phi(P^i_n) - \Phi(P^i_{n-1}) = U^i (X'_n - X'_{n-1}) + (V^{i'}_n - V^{i'}_{n-1})$

Let $X_n = (X'_n - X_0)$ and $V^i_n = (V^{i'}_n - V^{i'}_{n-1})$, we derive: $\Delta\Phi^i_n => U^i X_n + V^i_n$
We simply use the following generic formula to express how the initial scope stretches out to new scope: $\Delta\Phi^i => U^i X + V^i$
Let us assume the program P^i experienced 'c' times' change including add-on requests, and $\Delta\Phi^i_c$ denotes the scope deviation between Φ^i_c and Φ^i_c

1. first business scope change: $\Delta\Phi^i_1 => U^i X_1 + V^i_1$
2. second cumulative change: $\Delta\Phi^i_2 => U^i X_2 + V^i_2$
3. 'c'th scope change: $\Delta\Phi^i_c => U^i X_c + V^i_c$

generic equation we derived: $\Delta\Phi^i => U^i X + V^i$
cumulative equation: $\Delta\Phi^i_c => \Delta\Phi^i_1 + \Delta\Phi^i_2 + \ldots + \Delta\Phi^i_{c-1}$ as scope is accumulative
$X^i_c => X^i_1 + X^i_2 + \ldots + X^i_{c-1}$
$V^i_c => V^i_1 + V^i_2 + \ldots + V^i_{c-1}$

In pattern 1, as most conventional business analysis applies a "use case" approach in deriving its functions and non-functions, this issue requires further development. If we deal with each use case as a business component of pattern 3 and 4, which are component-based, we derive a generic equation:
$\Delta\Phi^i_c = \Delta\Phi^i_1 + \Delta\Phi^i_2 + \ldots + \Delta\Phi^i_{c-1}$
The equation actually makes no difference for which pattern it belongs to; we can easily identify from paperwork which one belongs to the business scope change and which one belongs to the add-on enhancement. However, the impact to P^n for each pattern is very different. In pattern 1, the program change caused by the scope change impacts the entire program (except the frozen part) no matter whether it is a change case or add-on case. First, we use the integration function $\int (U^i X + V) dx$ to represent the integration of technical effort against dimension 'x' which is associated with business scope change $\Delta\Phi^i$. Dimension 'x' is actually the mediation coefficient which can be a measurement of time, cost or man-days which need to reflect the effort spent on $\Delta\Phi^i$

We assume the first implementation from Φ^i_0 to Φ^i_1 ($\Delta\Phi^i_1$) consumes ΔP^i_1 technical effort, and the succeeding implementation is expressed as follows:
$\Delta P^i_1 = \Gamma(\Delta\Phi^i_0) = \Gamma((U^i X + V^i))$ where Γ is a linear transformation that projects $\Delta\Phi^i$ to ΔP^i_1
$\Delta P^i_2 = \int_{1,2} \Gamma((U^i X + V^i))\, dx = \Gamma((U^i X^2)/2 + V^i X)$
$\Delta P^i_3 = \int_{2,3} \Gamma((U^i X^2/2 + V^i X))\, dx = \Gamma((U^i X^3)/3 + (V^i X^2)/2)$
$\Delta P^i_n = \int_{n-1,n} \Gamma((U^i X^{n-1}/(n-1) + V^i X^{n-2}/(n-2)))\, dx = \Gamma((U^i X^n)/n + (V^i X^{n-1})/(n-1))$
$\Sigma\Delta P^i = \Delta P^i_1 + \Delta P^i_2 + \ldots + \Delta P^i_n$
$= \Gamma((U^i X^n)/n + ((U^i + V^i)X^{n-1})/(n-1) + ((U^i + V^i)(X^{n-2})/(n-2) \ldots + ((U^i + V^i)(X^2)/2 + ((U^i + V^i)(X)) + V^i)$
$= \Gamma((U^i X^n)/n + (U^i + V^i)((X^{n-1})/(n-1) + (X^{n-2})/(n-2) + \ldots + (X^2)/2 + (X)) + V^i)$
$= \Gamma((U^i X^n)/n + (U^i + V^i)\, \Sigma(X^m)/m + V^i)$ m=1..n-1

As the business changes are all equal in each pattern, all the values of U^i, n and m should be equal in each pattern. However, V^i and 'X' depend on the pattern we apply. Values of V^i in a reuse environment diminish as any new component is rare; thus we can merely find out how significant the 'X' impact is on the $\Sigma\Delta P^i$. Let us assume:
X = 2 means "parsing through whole scope and partial re-design is involved"

Copyright © 2007 Inderscience Enterprises Ltd.

X = 1.5 means "parsing through partial scope, partial aligned and few re-designed"
X = 1 means "scope is aligned and many new scopes are still non-reuse but cumulative"
X -> 0 means "scope is fully aligned and service can be reused in new scope"
By considering the convergence of the above equation, we use the following examples:
The original polynomial is like this: $f(x) = (\Sigma_{n=1,\infty} b_n (x-c)^n)(\Sigma_{n=1,\infty} d_n (x-c)^n)$
The differential and integration derived from the original are: $f'(x) = (\Sigma_{n=1,\infty} a_n n (x-c)^{n-1})$
From convergence theory, both of these series have the same radius of convergence as the original one. For the same reason, $\Sigma\Delta P^i$ converges to V^i when X approaches 0, and each of $\Delta P^i_1, \Delta P^i_2, \ldots, \Delta P^i_n$ has the same convergence when X approaches 0.

Now we benchmark the following patterns by using n = 5 (five times implementations) and let us assume $V^i = 10\%\ U^i$ means that the average add-on requirement takes 10% of the frozen program based on the equation $\Sigma\Delta P^i = \Gamma((U^i X^n)/n + (U^i + V^i)\ \Sigma_{m=1..n-1}(X^m)/m + V^i)$

Pattern 1 (non-component) X = 2
$\Sigma\Delta P^i = \Gamma((U^i 2^5)/5 + (U^i + V^i)\ \Sigma(2^5)/5 + V^i) = \Gamma(32\ U^i/5 + (U^i + V^i)\ (2+4/2+8/3+16/4) + V^i)$
$= \Gamma(32\ U^i/5 + 32/3\ (U^i + V^i) + V^i) = \Gamma(17.1\ U^i + 11.7\ V^i) = \Gamma(18.2\ U^i)$

Pattern 2 (component with limited alignment) X = 1.5
$\Sigma\Delta P^i = \Gamma(U^i\ (3.3/5) + (U^i + V^i)\ (1.5+2.25/2+3.3/3+4.9/4) + V^i)$
$= \Gamma(0.66\ U^i + 5.1\ (U^i + V^i) + V^i) = \Gamma(5.8\ U^i + 6.1\ V^i) = \Gamma(6.4\ U^i)$

Pattern 3 (component thin service layer) X = 1
$\Sigma\Delta P^i = \Gamma(U^i\ (1/5) + (U^i + V^i)\ (1+1/2+1/3+1/4) + V^i)$
$= \Gamma(0.2\ U^i + 2.1\ (U^i + V^i) + V^i) = \Gamma(2.3\ U^i + 3.1\ V^i) = \Gamma(2.6\ U^i)$

Pattern 4 (service based, fine and coarse grain mix)
4-3 thick service, fine grain aggregated to coarse grain, aligned to business X-> 0
$\Sigma\Delta P^i = \Gamma(V^i) = \Gamma(0.1\ U^i)$

4-2 thick service, fine and coarse grain mix X =(0,1]
$\Sigma\Delta P^i = \Gamma([0.1\ U^i, 2.6\ U^i))$ and $\Sigma\Delta P^i \rightarrow \Gamma(0.1\ U^i)$

4-1 service-based, fine-grained only X =(0,1] and (X=0 or X-> 1)
$\Sigma\Delta P^i = \Gamma([0.1\ U^i, 2.6\ U^i))$ and $\Sigma\Delta P^i < \Gamma(2.6\ U^i)$

Summary of $\Sigma\Delta P^i$ (for N=5, $V^i = 10\%\ U^i$)

Pattern	1(X=2)	2 (X=1.5)	3 (X=1)	41(X>0)	42(X>0)	43(X>0)
	$\Gamma(18.2\ U^i)$	$\Gamma(6.4\ U^i)$	$\Gamma(2.6\ U^i)$	$< \Gamma(2.6\ U^i)$	$> \Gamma(0.1\ U^i)$	$\Gamma(0.1\ U^i)$

The reason pattern 4 achieve minimum effort in service implementation is as an enriched services layer produces coarse-grained services which are aggregated from fine-grained components. Service dimensions, therefore, become an easily configured soft area for the provision of a full range of services. Figure 1 gives a full picture in presenting the thick

Copyright © 2007 Inderscience Enterprises Ltd.

service layer. The notable strategies and characteristics of interoperability include: a universal schema and metadata in the information layer; semantic mediation between business, service and component; interoperability between three repositories; and business decomposition in the upper layer. Among these, the determination of the service atomic component is the key for moving component-based architecture into semantic service. We are unable to merely rely on the configuration of the component in achieving dimensional service, as it does not make too much sense in semantic mediation within enterprise integration. However, we certainly need to apply component-centric design to fulfill interoperability requirements. The reasons for this are: 1) The service atomic component is the mediation unit between business and IT as this atomic element loosely binds business and IT terminology in a common view; only through this single point design do we have the opportunity to align enterprise architecture into interoperability; 2) From this single point design we can validate associated metadata in the repository for their analysis in noun, verb and attribute as an atomic element while using business semantic terms to mediate pre-built component sets which have been configured for that service; 3) The service component can be easily built, revised or aggregated to achieve any service change caused by business change because most of the physical component design is an atomic level function. The mechanism of interfacing and parameter transfer are all keys to how each component performs; therefore, the mechanism of the "integrated functions" should not be vaporized, rather they need to be stable and durable. This means the virtual service which is mounted on the supporting components should effectively apply metadata in the repository to mediate other services in provisioning a service dimension; once this service is validated, the metadata which holds the mediation information should be stored in a knowledge-based repository for reuse or revision; 4) The service layer is an easily configurable area which we can leverage to the scenarios of cache memory computing in mediating application and database storage. The syntax, logic, semantic and execution plans such as security features are all under metadata control; these are stored in repository and ready for configuration.

When we reviewed the results of our experiments in relation to the behavior of these patterns by using conventional parsing and a full-table scan in swapping and paging, we found all to be similar to pattern 1. The scenario of implementation point parsing was used through the whole program, as any minor change may cause a high impact. Patterns 3 and 4 are leveraged to semantic mediation. By using metadata mediation and semantic integration through cache memory computing, the physical read drops to 3.3% and the response time was cut to 4.3% when using conventional hash comparing and joining. Semantic mediation is the key in this enhancement for the following reason:

Hash computing consumes cache memory by using a full-table scan and swapping in/out the full tables. The huge volume table access, storage I/O and swap in/out cause inefficient computing, as element read-in can cause mass changes in a table scan. The 33 times physical read of hash comparing rather than semantic mediation makes this evident. This is mostly caused by the full-table scan due to weak semantic strategy; the high-volume data takes whole memory and recursively retrieves the database by swapping in/out. Comparatively, semantic computing uses cache memory in cataloging and indexing to focus on the mediator, thus the scope becomes small and the semantics are fully utilized in the mediation between "mediator" and data. By taking this scenario to semantic service implementation, we derived the following equation:

$$\Delta P = \Gamma(\Delta\Phi) = \Gamma(\Sigma\,\Phi_n(C_p) - \Sigma\,\Phi_{n-1}(C_q)) \,!= \Gamma(\Sigma\,\Phi_n(C_p)) - \Gamma(\Sigma\,\Phi_{n-1}(C_q))$$

which means even in a component-based environment, we have difficulty in making effective progress in technology by using the scope deviation between current and previous ones.

Characteristic Mediation and Syntax as an Atomic Element

A thick service layer enriches the 'm' value when a component space stretches out, thus

Copyright © 2007 Inderscience Enterprises Ltd.

contributing a higher 'ε' value which is the dimensional services space. However, we need to optimize the relationship between 'ε' and 'k' which we call "the space of effective dimensional services." As although large 'ε' contributes a variety of services and higher service coverage, the side effect is that it may cause higher complexity in the service ontology and a duplicate foundation may exist in common practice. Let "->" denote mediation and "|=>" denote aggregation; a duplicate example is as follows:

$[C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_5 \rightarrow C_7] \models> [S_{p1} \rightarrow S_{t1} \rightarrow S_{t2} \rightarrow S_{p2}] \models> S^p$

$[C_1 \rightarrow C_2 \rightarrow C_3 \rightarrow C_6 \rightarrow C_9] \models> [S_{q1} \rightarrow S_{t1} \rightarrow S_{t2} \rightarrow S_{q2}] \models> S^q$

In this example, as "$C_1 \rightarrow C_2 \rightarrow C_3$" and "$S_{t1} \rightarrow S_{t2}$" repeated in the many service routes which need to be replaced by a coarse-grained component or service which slightly compresses and downsizes the 'k' value. However, it consequently improves the maturity of each dimension in the 'ε' service dimensions. In contrast to 'ε', as the 'k' value may already include those repeatable "routes" in many service dimensions, we need to consolidate the common area and aggregate them into coarse-grained service. From a quality perspective, this is why the "maturity" of the service route (which has been aggregated from atomic elements) very often can be of equal importance to the "variety" of service combination from the quantity perspective. The optimization, therefore, needs both fine-grained and coarse-grained services in order to optimize the relationship between 'ε' and 'k' from 'm', which is a fixed value of the atomic element in component space.

H. Lee, in his domain analysis, proposes that the extraction of verbs and nouns for identifying domain objects needs to be followed by sets of use case scenarios. These are formulated so that the analysis of the scenario and the extraction of noun and verb form the common components of process, object and interface as domain assets. For micro component design, the semantic analysis requires a metadata repository to store noun, verb and their attributes to construct the service mediation foundation. This construction needs both bottom up and top down approaches to extract noun, verb and attribute to create this repository. In the semantic repository, the noun list usually stores customer name, ID, phone, account number and status, while the verb list typically stores open account (type), check non-exist (db), create account (type) and credit check (code). Once the semantic analysis work is completed and the associated verb, noun and attribute lists are produced and stored in the semantic repository, common objects need to be validated and registered as common assets before further development. Asset validation is a process in the vertical (downward) direction for making duplicate and syntax checks. The strategy of "enterprise vertical integration" we introduced was based on the processes which build up these repositories in the first phase as well as the methods in achieving semantic interoperability between repositories in the second phase. Both service element mediation and common service are crucial for the second phase; the fact that all of the organizations share the common service means they are all managed under the same rules which were developed using the top-down approach. As all of the organizations share a common protocol to communicate their semantic rules, the integration effort can be eliminated and interoperability can be enhanced using this topology once the semantic communication has been fully developed.

Therefore, the determination of the service atomic component is the key for moving component-based architecture into semantic service. We are unable to merely rely on the configuration of the component in achieving dimensional service, as it does not make too much sense in semantic mediation within enterprise integration. However, we certainly need to apply component-centric design to fulfill interoperability requirements. The reasons for this are: 1) The service atomic component is the mediation unit between business and IT as this atomic element loosely binds business and IT terminology in a common view; only through this single point design do we have the opportunity to align

Copyright © 2007 Inderscience Enterprises Ltd.

enterprise architecture into interoperability; 2) From this single point design we can validate associated metadata in the repository for their analysis in noun, verb and attribute as an atomic element while using business semantic terms to mediate pre-built component sets which have been configured for that service; 3) The service component can be easily built, revised or aggregated to achieve any service change caused by business change because most of the physical component design is an atomic level function. The mechanism of interfacing and parameter transfer are all keys to how each component performs; therefore, the mechanism of the "integrated functions" should not be vaporized, rather they need to be stable and durable. This means the virtual service which is mounted on the supporting components should effectively apply metadata in the repository to mediate other services in provisioning a service dimension; once this service is validated, the metadata which holds the mediation information should be stored in a knowledge-based repository for reuse or revision; 4) The service layer is an easily configurable area which we can leverage to the scenarios of cache memory computing in mediating application and database storage. The syntax, logic, semantic and execution plans such as security features are all under metadata control; these are stored in repository and ready for configuration.

Producing a Service Map by Component Evaluation

In the notation on service mediation, we pointed out that enterprise architecture consists of common services and project specific services; although common services is our enterprise priority, both categories are mandatory in a project and require a variety of service dimensions to take the business vision into realization. Therefore, when a business goal is aligned with dimensional services, each dimension can be a composite service which consists of many components within different service categories. These dimensions need patterns to fulfill their technical implementation and, as the micro process in the service pattern needs a map to derive the dimension, the service map needs to be the discipline of pattern design and serves as a modeling tool for dimensional service. To demonstrate how the service map is created, let $\Phi(C_n)$ denote the matrix of Cartesian space from the component comparison for component analysis; where θ_{ij} is the comparison between adjacent components C_j against C_i

$$\Phi(C_n) = \begin{bmatrix} \theta_{11}\ \theta_{12} \cdots \theta_{1n} \\ \cdots\cdots \\ \theta_{n1}\ \theta_{n2} \cdots \theta_{nn} \end{bmatrix}$$

As service mediation needs both top down and bottom up approaches which rely on component mediation in different layers, we apply eigenvalue techniques to derive a service pattern as our objective is to produce a service map through component pair-wise comparison as well as to iterate the dimension of service for pattern design. To achieve this goal, we first review the general principles of eigenvalue and bring it into our pattern iteration scenarios:

Equation 1. $\Phi(C) = \lambda I$ where λ is the eigenvalue

Equation 2. When we deduct this from a regular matrix X^n it gives a new matrix $X^n - \lambda I$
$X^n - \Phi(C) = X^n - \lambda I$ which we call X^n the deducting vector

Equation 3. We derive the determinant $|X^n - \lambda I| = 0$

Equation 4. By multiplying the same row of X^n, we derive the eigen vector S^n_i
$(X^n - \lambda I) \times S^n_i = 0$ which we call the service dimensional vector

Equation 5. Since $I = 1$ we can write the general expression $X^n \times S^n = \lambda S^n$

Equation 6. For the largest eigenvalue $\lambda^{(max)}$ $X^{(max)\,n} \times S^n = \lambda S^n$

Copyright © 2007 Inderscience Enterprises Ltd.

Equation 7. If λ means the possibility of a service component in mediation of the components into a dimensional service, then according to the transition probability matrix of the Markov chain, we set $\lambda^{(max)} = 1$ and derive $X^{(max)\,n} \times S^n = S^n$. Now let us practice the process for taking the pair-wise comparison to derive the service map:

From equation 1, we assume the following general matrix of $\Phi(C_n)$

$$\Phi(C_n) = \begin{bmatrix} 0 & 1 & 0.5 \\ 1 & 0 & 0.5 \\ 1 & 0.5 & 0 \end{bmatrix} \qquad \Phi(C_n) - \lambda I = 0$$

From equation 2, we derive $X^n - \Phi(C) = X^n - \lambda I$

$$\text{Let } X(C_n) + \Phi(C_n) = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

From equation 3, $|X^n - \lambda I| = 0$

$$X(C_n) = \begin{bmatrix} 1 & 0 & 0.5 \\ 0 & 1 & 0.5 \\ 0 & 0.5 & 1 \end{bmatrix} \begin{bmatrix} 1-\lambda & 0 & 0.5 \\ 0 & 1-\lambda & 0.5 \\ 0 & 0.5 & 1-\lambda \end{bmatrix} = 0$$

$$(1-\lambda)^3 - 0.25(1-\lambda) = 0 \qquad \lambda_1 = 1, \lambda_2 = 0.5$$

Where λ_1 is the principal eigenvalue. When we take these eigenvalues to transform the regular vector into service dimensions: From equation 4 $(X^n - \lambda I) \times S^n_i = 0 \times I$, we then apply $\lambda_1 = 1$ to derive the eigen vectors S^n_i

$$\begin{bmatrix} 1-1 & 0 & 0.5 \\ 0 & 1-1 & 0.5 \\ 0 & 0.5 & 1-1 \end{bmatrix} \times \begin{bmatrix} S^1_1 \\ S^1_2 \\ S^1_3 \end{bmatrix} = 0 \times I$$ This derive $S^1_2 = S^1_3 = 0$ and S^1_1 can be any value

$$\begin{bmatrix} 1-0.5 & 0 & 0.5 \\ 0 & 1-0.5 & 0.5 \\ 0 & 0.5 & 1-0.5 \end{bmatrix} \times \begin{bmatrix} S^2_1 \\ S^2_2 \\ S^2_3 \end{bmatrix} = 0 \times I$$ This derive $S^2_1 = S^2_2 = -S^2_3$

As shown in Figure 2, $\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$ is one of the valid eigenvector for eigenvalue $\lambda_2 = 0.5$ in S^n

Component Evaluation Case Study in Service Mediation

The above example is a theoretical iteration and does not have a business scenario as its goal. We will now use a case study to apply the theory into a real scenario. As is generally known, inter-component mediation is achievable by the evaluation of a pair-wise comparison; by means of this practice we have an opportunity to create visibility of the dimensional service in the iteration process.

Copyright © 2007 Inderscience Enterprises Ltd.

As formulated by Saaty (1980), the Analytic Hierarchy Process (AHP) is an intuitive-judgmental multiple criteria approach to decision making. According to Saaty, a number of decision elements such as the virtual layer and the element layer are arranged in a hierarchical structure.

Before constructing the service alignment map, we assign the following symbols to represent the relationship between adjacent components which are C_i and C_j. Let θ (C_{ij}) denote the comparison and degree of dependency from C_j to C_i:

'$C_i \overline{\top} C_j$' denotes that C_i and C_j have a binding relationship in C_{ij} service provisioning

'$C_i \top Cj$' denotes that C_i and C_j have a conditional relationship in C_{ij} service provisioning

'$C_i \mid C_j$' denotes that C_i and C_j have no relationship and never join

'&' denotes co-existence

'~' denotes either; and

'=>' denotes derivation

			Rating
$(C_1 \overline{\top} C_2) \& (C_2 \overline{\top} C_3)$	=>	$(C_1 \overline{\top} C_3)$ strong mediation	1.0
$(C_1 \overline{\top} C_2) \sim (C_2 \overline{\top} C_3)$	=>	$(C_1 \top C_3)$ weak conditional mediation	0.5
$(C_1 \top C_2) \& (C_2 \top C_3)$	=>	$(C_1 \top C_3)$ strong conditional mediation	0.7
$(C_1 \top C_2) \sim (C_2 \top C_3)$	=>	$(C_1 \mid C_3)$ weak conflict	0.3
$(C_1 \mid C_2) \& (C_2 \mid C_3)$	=>	$(C_1 \mid C_3)$ strong conflict	0
$(C_1 \mid C_2) \sim (C_2 \mid C_3)$	=>	$(C_1 \mid C_3)$ strong conflict	0.1
$(C_1 \overline{\top} C_3) \& (C_3 \overline{\top} C_4)$	=>	$(C_1 \overline{\top} C_4)$ strong mediation	1.0
$(C_2 \overline{\top} C_3) \sim (C_3 \overline{\top} C_4)$	=>	$(C_2 \top C_4)$ weak conditional mediation	0.5
$(C_1 \mid C_3) \& (C_3 \mid C_4)$	=>	$(C_1 \mid C_4)$ strong conflict	0

In the strong mediation scenario of $(C_1 \overline{\top} C_2) \& (C_2 \overline{\top} C_3)$, C_2 plays the mediator role in mediating C_1 and C_3, and other conditional mediation are all candidate mediators in component mediation; if we can clearly set the conditions for these mediations to become a stronger mediation, we have the opportunity to aggregate these mediators into coarse-grained services. Based on these conditions, the portfolio of coarse-grained services will construct the foundation for both common service and multi-dimensional services.

Before evaluation, our top-down approach uses the above notation to hold the initial mapping between the virtual and element layer. In the following paragraph, we use a project example to demonstrate how to create the service map in order to model the service pattern. The project, called RCSA (Rent Car Service Agent), basically consists of three business blocks (reservation, rental and payment) and a newly-added feature (logistics and batch process). Each business component consists of several service components and associated physical components. During the domain decomposition, we produced the semantic element in the atomic form consisting of noun, verb and attribute which were stored in the semantic repository for further development. Basically, our physical component uses the semantics for its micro process design in interfacing and metadata mediation. The domain decomposition provides a rough direction in identifying what service component the project requires, which service components are common components that can be reused, and how the newly-built components in the project work in conjunction with common components. The enterprise modeling is a required process in our common component analysis (Vernadat F., 1997).

The newly-launched program, DSPO (Dynamic Switch to Pre-paid Option), allows frequent RCSA customers to use a credit card to collect their travel bonus points and

Copyright © 2007 Inderscience Enterprises Ltd.

automatically switches to the VIP account for the pre-paid option which combines the credit card and bonus point accounts. The newly-launched program identifies individual components from three service categories including reservation, rent and payment, and uses pair-wise comparison to evaluate their relative weight and significance projected on each of the related services. From the eigenvalue base equation, $AX = \lambda X$ where X is the eigenvector and λ is the eigenvalue, we derive the equation which follows.

$$\begin{bmatrix} a_{11}-\lambda & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22}-\lambda & \cdots & a_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kk}-\lambda \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Pair-wise comparison of components against a service dimension is based on a 1-9 score that commences by comparing the relative importance of one component versus another on the same level. The evaluation process involves: 1) pair-wise comparisons of components; 2) pair-wise comparisons of the service dimension; and 3) integration of comparisons into the business objective. The evaluation shown in Figure 3 gives an impression for how pair-wise comparison derives the relative weight of each component in relation to a service and how the service component is composed from component elements with different weights; these provision a dimensional service with their individual weight. The component and domain analysis is shown in Appendix 1 and the relative weight result is shown in Appendix 2. In our evaluation we derived three level iterations: 1) how the component mediates to achieve a service component; 2) how the service map is created through service components; and 3) how the dimensions achieve a business goal.

Conclusions

This study proposes our semantic mediation strategy should not be started from the system or application level, rather our concerns were initiated from micro process. We believe that without this step we will not be able to cope with business change and will consequently lose IT service competition. To comply with service mediation principles, there are three significant issues which need immediate action:

First, from a macro architecture view, enterprise loses governance in semantic interoperability due to misalignment between the business, service and technical components. In order to overcome the issues we proposed an ontology of component-based mediation, in which both top-down and bottom-up approaches were proposed. We believe the ontology requires a robust mechanism in semantics base architecture; this will need a "big picture" that encompasses the common component, semantics and metadata repository into a state-of-art interoperability and this is what we mean the "skeleton" of common service.

Second, from a micro architecture view, semantic mediation has been misinterpreted by several activities and needs a new direction in order to fix common issues. The fact is, in current stage, the scope of interoperability has not been stretched out to the global scale, thus the mediation has been limited to only conjunctions of the architecture layer such as application interface, system protocol, and metadata messaging. We need to realize that both component analysis and mediation synthesis are all required before dimensional service is identified and validated, this is also mandatory for those common services to construct the enterprise skeleton and the general component in project service.

Third, from the view of semantics ontology, to create the alignment visibility and simultaneously, maintain a semantics strategy requires a repository to hold the metadata of mapping matrix such as, service dimensions, service patterns and mediation strategy. This guarantees the minimum re-design and less effort spent in each change.

Copyright © 2007 Inderscience Enterprises Ltd.

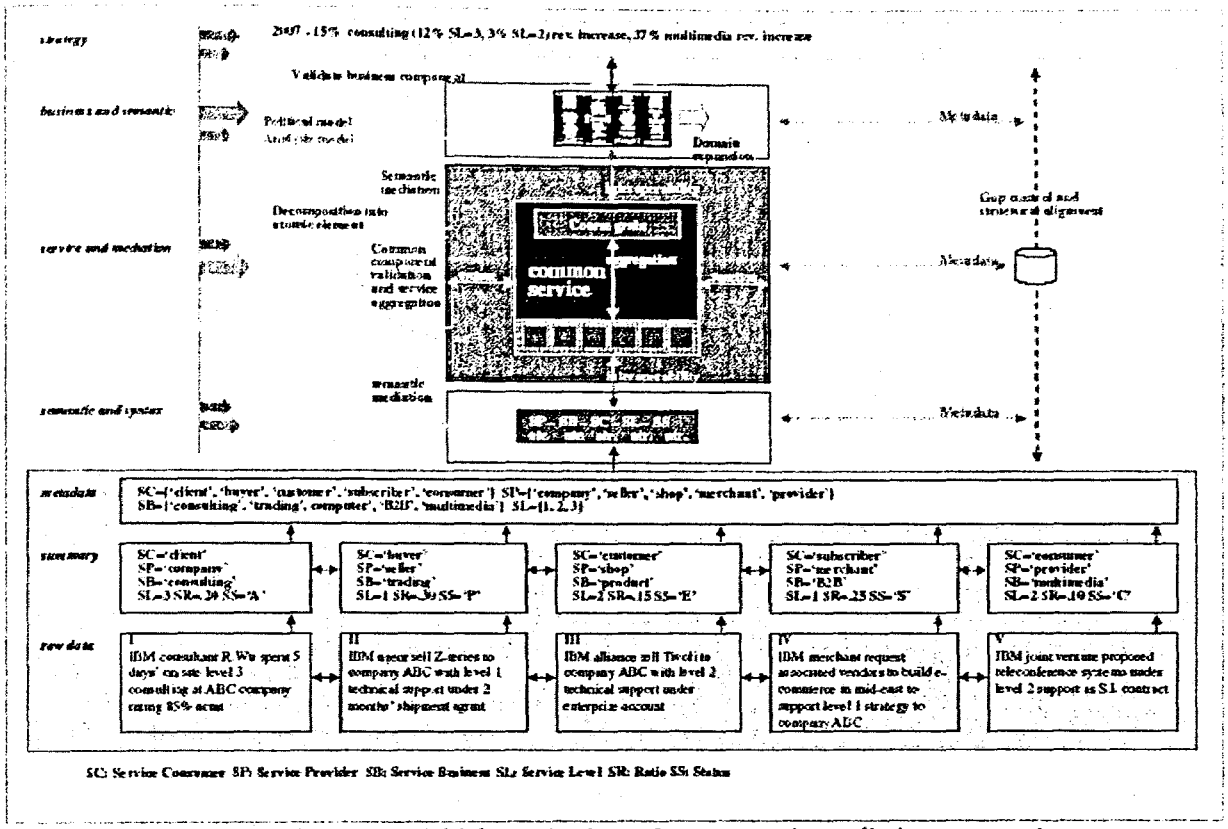


Figure 1 Full picture of thick service layer from semantic mediation perspective

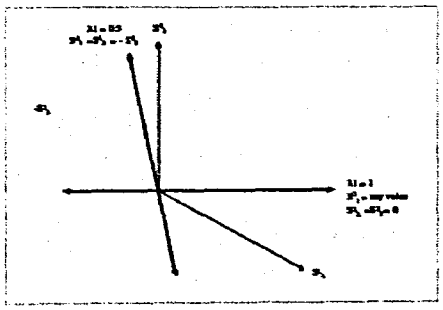


Figure 2 Dimensional service drived from eigenvalue

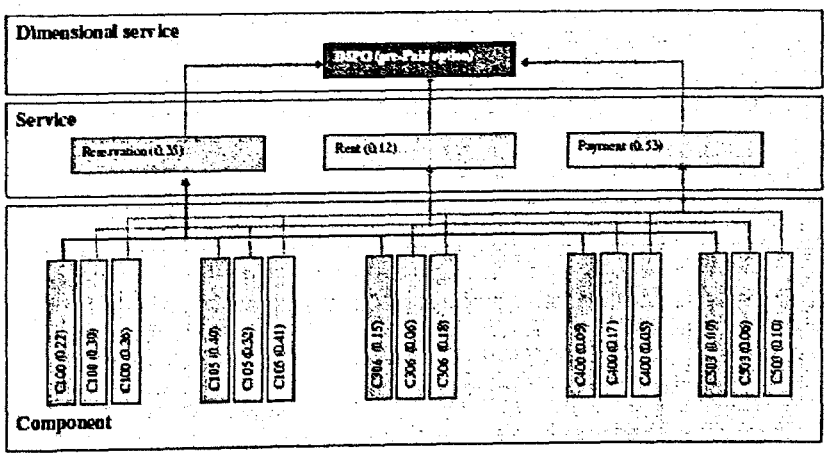


Figure 3 AHP in DSPO dimensional service with relative weight of composition

Copyright © 2007 Inderscience Enterprises Ltd.

Reference

Gruninger M., (2005), "Model-theoretic approaches to semantic integration", Dagstuhl Seminar Proceedings 04391 Semantic Interoperability and Integration 2005

Wu, R., (2007) - "Semantic Metadata in Enterprise Integration", IEEE Southeast 2007 conference, pp 233-236

Sicilia, M., (2006), "Metadata, semantics, and ontology: providing meaning to information resources", *Int. J. Metadata, Semantics and Ontologies, Vol. 1, No. 1, 2006, pp 83-86*

Sebastian Ryszard Kruk, Adrian Mocan and Brahmananda Zaremba, (2005), "Building Semantic Web Services Infrastructure for Digital Libraries", DERI-Lion project, and by Gdansk University of Technology, and by the European Commission under the projects DIP, Knowledge Web, SEKT, SWWS, and Esperonto, 2005 (working paper)

Pedrinaci C., Moran M., and Norton B, (2006) , "Towards a Semantic Event-Based Service-Oriented Architecture", EU Integrated Projects, DIP FP6-507483 and SUPER FP6-026850

Alessio Gugliotta, Vlad Tanasescu, John Domingue, Rob Davies, Leticia Gutiérrez-Villarías, Mary Rowlatt, Marc Richardson and Sandra Stinčić, (2006), "Benefits and challenges of applying semantic web services in the e-Government domain", DIP (FP6 – 507483), an Integrated Project funded under the European Union's IST programme. 2006

Moulton A., Madnick M., and Siegel M., (2003), "Context Interchange Mediation for Semantic interoperability and Dynamic Integration of Autonomous Information Sources in the Fixed Income Securities Industry", No 4404-02 Working papers from MIT Sloan School of Management

Vckovski A., Brassel K. and Schek H., (1999), "Interoperating Geographic Information Systems", Second International Conference, INTEROP March 1999 Computer Science 1580 Springer

Mishra A., Ripley M., and Gupta A., (2002), "Using Structural Analysis to Mediate XML Semantic Interoperability", MIT Sloan Working PP 4345-02 No. 1024-05 February 2002

Chan, L. and Zeng, M., (2006), "Metadata Interoperability and Standardization – A Study of Methodology Part I", D-Lib Magazine June 2006

Kashyap V. and Sheth A., (1996), "Semantic and Schematic Similarities between Database Objects: A Context-based approach", VLDB Journal: Very Large Data Bases

Park J. and Ram S., (2004) "Information systems interoperability: What lies beneath?", ACM Trans. Inf. Syst. 22(4) 595-632

Panfilis S. and Berrel A., (2001), "Open issues and concerns on Component Based Software Engineering", CBSEnet publication IST-2001-35485

Vernadat F., (1997), "Enterprise Modelling Languages", ICEIMT 1997 Enterprise Integration - International Consensus

Sheth A., (2005), "From Semantic Search & Integration to Analytics", Dagstuhl Seminar Proceedings 04391 Semantic Interoperability and Integration LSDIS lab, University of Georgia

Qin, J. and Prado, J.C. (2006), "The semantic and syntactic model of metadata", Série Comunicação da Informacao Digital 2006 Vol. 4 (pp. 143-156)

Qin, J. and N. Hernandez. (2004), "Ontological representation of learning objects: building interoperable vocabulary and structures", ACM Press New York, May 17-22, 2004, 348-349

Davis L., Gamble R., Payton J., Jónsdóttir G. and Underwood J. (2001), "A Notation for Problematic Architecture Interactions", 9th Int'l Symposium of Software Engineering (ESEC/FSE-9) pp. 132-141

Cambrin D., Demeule L., Dumedah G., Hoare D. And Simmonds K. (2004), "Sub-Surface Data Management", Geography 455 Advanced Issues in GIS March 2004

Copyright © 2007 Inderscience Enterprises Ltd.

Appendix

Appendix I Component analysis for DSPO

Service Hierarchy

Reserve Vehicle	S^1_{10}	$C_{100}\overline{+}$
• Check Rate	S^1_{11}	$C_{100}\overline{+}C_{101}$
• Get Location (pick-up/drop-off)	S^1_{12}	$C_{100}\overline{+}C_{102}$
• Get Date/time (pick-up/drop-off)	S^1_{13}	$C_{100}\overline{+}C_{103}$
• Choose Vehicle	S^1_{14}	$C_{100}\overline{+}C_{104}$
• Get Options Information	S^1_{15}	$C_{100}\overline{+}C_{106}$
• Check vehicle Availability	S^1_{16}	$C_{100}\overline{+}$
• Offer Rates for Selection	S^1_{17}	$C_{100}\overline{+}C_{105}$
• Confirm Rental Information	S^1_{18}	$C_{100}\overline{+}C_{107}$
• Get Customer Information	S^1_{19}	$C_{100}\overline{+}C_{104}$
Get Payment Information	S^1_{20}	$C_{101}\overline{+}$
Confirm Reservation	S^1_{30}	$C_{203}\overline{+}C_{207}\overline{+}C_{204}$
Create Reservation	S^1_{40}	$C_{204}\overline{+}C_{702}$
Locate Reservation	S^1_{50}	$C_{205}\overline{+}C_{702}$
Modify Reservation	S^1_{60}	$C_{206}\overline{+}$
Make Reservation	S^1_{80}	$C_{203}\overline{+}$
Cancel Reservation	S^1_{90}	$C_{208}\overline{+}C_{108}$
Check-out Vehicle	S^2_{10}	$C_{301}\overline{+}$
• Sign-out Vehicle from Lot	S^2_{11}	$C_{302}\overline{+}C_{502}$
• Create Rental Agreement	S^2_{12}	$C_{302}\overline{+}C_{702}\overline{+}C_{304}$
Check-in Vehicle	S^2_{20}	$C_{305}\overline{+}$
• Locate Rental Agreement	S^2_{21}	$C_{304}\overline{+}C_{702}$
• Process Return Information	S^2_{22}	$C_{205}\overline{+}C_{108}$
• Void Rental Agreement	S^2_{24}	$C_{307}\overline{+}C_{108}$
• Modify Rental Agreement	S^2_{23}	C_{306}
Process Payment	S^3_{10}	$C_{401}\overline{+}C_{501}$
Process Refund	S^3_{20}	$C_{402}\overline{+}C_{502}$
Pre-paid Option (new)	S^3_{30}	$C_{403}\overline{+}C_{503}$
Vehicle Trace	S^4_{10}	$C_{601}\overline{+}$
Vehicle Status	S^4_{20}	$C_{602}\overline{+}$
Execute nightly Batch	S^5_{10}	$C_{900}\overline{+}C_{401}\overline{+}C_{402}\overline{+}C_{601}$

Domain Semantic Analysis

Vehicle			x
Refund			x
Batch			x
Data Object			x
Payment		x	
Agreement		x	
Reservation		x	
Customer	x	x	x
Information	x	x	x
Sales	x		
Agent	x		
Sign-in			x
Sign-out			x
Confirm			x
Report			x
Reserve	x		
Check	x		
Create	x		
Locate	x		
Process	x		
Modify	x		
Nullify	x		
Cancel	x		
Hold	x		
Trace	x		
Transport	x		
Rental Options			x
Availability			x
Vehicle Type			x
Rate			x
Location	x		
Date/time	x		
Confirmation	x		
Cancellation	x		
Modification	x		
Status	x		

Component Analysis

Component Function	ID	Description
View Vehicle Pool Availability (VPA table)	C_{100}	DM (S: Vehicle Pool table & auto generate ext_partition_para for access option
Confirm Rental Status (CRO table)	C_{200}	DM (S/U/I) Rental Status table & auto generate ext_partition_para for access option
Vehicle Check in-out root program	C_{300}	Sync Rental Status table & Vehicle Pool table by triggering CRO UC key
Payment root program	C_{400}	DM (S/U/I) Client Mgt table & auto generate ext_partition_para for access option
Logistics root program	C_{500}	DM (S/U/I) Vehicle Mgt table
Reporting root program	C_{600}	DM (S) & auto generate ext_partition_para for access option
Maintain Rental Agreement RA	C_{700}	DM (S/U/I) Rental Agreement table
Batch VRC-01 (Vehicle, Rental & Client)	C_{900}	Nightly Batch process scheduled by cron job
Select Rate Range (VPA RR key)	C_{101}	DM (S) Vehicle Pool table by executin ext_partition_para RR for access option
Select Location Map (VPA LM key)	C_{102}	DM (S) Vehicle Pool table by executin ext_partition_para LM for access option
Select Time Range (VPA TR key)	C_{103}	DM (S) Vehicle Pool table by executin ext_partition_para TR for access option
Select Vehicle Type (VPA VT key)	C_{104}	DM (S) Vehicle Pool table by executin ext_partition_para VT for access option
Select Pre-paid Price Range (VPA PR key)	C_{105}	DM (S) Vehicle Pool table by executin ext_partition_para PR for access option
Select Insurance Option (VPA IO key)	C_{106}	DM (S) Vehicle Pool table by executin ext_partition_para IO for access option
Select Get Options (VPA GO key)	C_{107}	DM (S) Vehicle Pool table by executin ext_partition_para GO for access option
Report Generation for Selection criteria	C_{108}	DM (S) Vehicle Pool table & execute C_{600}
Select Customer Info (CRO CI key)	C_{201}	DM (S) Confirm Rental table by executin ext_partition_para CI for access option
Select Payment Info (CRO PI key)	C_{202}	DM (S) Confirm Rental table by executin ext_partition_para PI for access option
Update Confirmation (Update CRO UC)	C_{203}	DM (U) Confirm Rental table by executin ext_partition_para UC to activate access
Create Reservation (CRO UC='1')	C_{204}	DM (I) Confirm Rental table by executin ext_partition_para UC for access option=1
Make Reservation (CRO UC='2')	C_{205}	DM (U) Confirm Rental table by executin ext_partition_para UC for access option=2
Locate Reservation (CRO UC='3')	C_{206}	DM (U) Confirm Rental table by executin ext_partition_para UC for service allocation
Hold Reservation (CRO UC='9')	C_{207}	DM (U) Confirm Rental table by executin ext_partition_para UC to deactivate access
Cancel Reservation (CRO UC='0')	C_{208}	DM (U) Confirm Rental table by executin ext_partition_para UC to terminate service
Vehicle Check-out (RA CO='1')	C_{301}	Update Vehicle Pool Status table & auto generate RA CO script for update option=1
Sign-out Vehicle (VPS SV='1')	C_{302}	Update Vehicle Pool Status table & auto generate VPS SV script for update option=1
Check-in Vehicle (RA CO='2')	C_{303}	Update Vehicle Pool Status table & auto generate RA CO script for update option=2
Locate Rental Agreement (VPS SV='2')	C_{304}	Update Vehicle Pool Status table & auto generate VPS SV script for update option=2
Process Return Info (VPS SV='3')	C_{305}	Update Vehicle Pool Status table & auto generate VPS SV script for update option=3
Modify Rental Agreement (new)	C_{306}	Update Vehicle Pool Status table & auto generate VPS SV script for update option=9
Void Rental Agreement (VPS SV='0')	C_{307}	Update Vehicle Pool Status table & auto generate VPS SV script for update option=0
Payment Mgt CM & PM Trace	C_{401}	DM (S) Client Mgt table & Payment Mgt table
Payment Mgt CM & PM Update	C_{402}	DM (U/I) Client Mgt table & Payment Mgt table
Payment Mgt CM & PM Batch	C_{403}	C_{900} Batch interfacing
Trace FM table	C_{601}	DM (S) Fleet mgt table by executin ext_partition_para for access option
Status Reporting	C_{602}	DM (S) Fleet mgt table by executin ext_partition_para for C_{600} printing
Logistics interface	C_{603}	DM (S/U/I) Fleet mgt table by executin ext_partition_para for C_{500}

Appendix 2 Relative weight in AHP analysis

component 1	0.3 3 3 3	22 %
component 2	3 7 5	49 %
component 3	3 3	15 %
component 4	0.3	5 %
component 5		9 %
component 1	1 3 3 5	39 %
component 2	3 3 4	32 %
component 3	0.3 0.5	6 %
component 4	3	17 %
component 5		6 %
component 1	0.5 2 5 3	26 %
component 2	3 5 4	41 %
component 3	4 3	18 %
component 4	0.3	5 %
component 5		10 %
Payment	Reservation Rent Payment	relative weight
Reservation	3 0.5	35 %
Rent	0.3	12 %
Payment		53 %

Copyright © 2007 Inderscience Enterprises Ltd.

T2

T2: An industry case study of micro component design and semantic mediation

Publication: IJMSO Journal 2007

International Journal of Metadata, Semantics and Ontologies (IJMSO)
(http://www.inderscience.com/browse/index.php?journalCODE=ijmso)

Publication Contact: Prof. Miguel-Angel Sicilia
Editor in chief, IJMSO Inderscience
Computer Science, Technical School of Informatics
University of Alcal, Spain
msicilia@uah.es

From:	"Miguel-Angel Sicilia" <msicilia@uah.es> Add to Address Book Add Mobile Alert
To:	"'Raymond Wu'" <rwu_academic@yahoo.com>
Subject:	RE: RE: Submission for your review for publication purpose - IJMSO
Date:	Thu, 31 May 2007 13:03:58 +0200

Dear Raymond,

The papers are accepted and scheduled for publication, so only minor c formatting changes will be required as they go through the publishing process. You can safely mention them as definitely accepted in the record of
your Ph.D. dissertation, even though the final publication date is still
subject to uncertainty.

* Semantic Mediation Modeling in the Service Component Design

* An industry case study of micro component design and semantic mediation

Best regards,
Miguel-Angel

International Journal of Metadata, Semantics and Ontologies ISSN: 1744-2621

An Industry Case Study of Micro Component Design and Semantic Mediation

Raymond Wu

IBM Software Group
Email: rwu_academic@yahoo.com

Abstract: *This paper is a case study of enterprise integration in financial industry which consists of two parts, first part is component based design, and second part is semantic mediation, it explain why componentization and semantics are both critical and their interference in enterprise integration. Our current research has been extended to service virtualization, which is the final stage in service provisioning, this research paper reviews our previous work, positioning component and semantics in the life cycle, and uses examples in the industry to demonstrate component-based implementation and semantics mediation. To investigate the fundamental mechanism for the further understanding of the meaning of semantic mediation, we execute benchmarking by using both empirical experiments and theoretical modeling. Our observations show that a significant enhancement can be achieved by using the mediation strategy in semantic metadata.*

Keywords: semantic mediation, financial industry, enterprise integration, component benchmarking, services

Biographical notes

Raymond Wu is the IT architect currently working with IBM Software Group, his expertises include, enterprise architecture, integration, metadata, service transformation, financial, telecom and database design. Previously he worked with Oracle, Citibank and Vodafone as enterprise architect.

Introduction

The IT industry in financial sector differentiate itself from other sectors in that higher security requirements are mandatory, and interbanking service need agility in mediating different parties into a common service infrastructure. A typical financial project consists of domain implementation, architecture and shared services. The domain implementation group is divided into several teams to handle various LOBs (lines of business) as each team will have different areas of business domain knowledge; each team may in fact handle several business lines (i.e., use cases). It's very often the case that the individual LOB team doesn't have adequate concepts and communication to determine the common components, which include common services, common interface and standard data access and data models. We would like to review and list these common issues in enterprise integration, along with our proposals for resolution.

i) Too many integration points cause implementation silos

ii) Data fragmentation with duplicate process without aggregation strategy (Curtis et al, 1999)

Copyright © 2007 Inderscience Enterprises Ltd.

iii) Common area is invisible and unable for reuse in a shared model
iv) Lack of priority and focus of strategy slow down business responsiveness
v) Weak enterprise interoperability causes high risk in industry transformation

This scenario is common in the financial industry, one of the example is the conventional credit card service doesn't maintain a robust fraud detection policy at the account level, so many transactions become invalid, the agility and responsiveness of making a proper change is still weak, and many consumers suffer the serious issue that their transaction needs to be rolled back. For this reason, we launched the add-on service component of Fraud Policy service to replace the conventional practice and dynamically retrieve cardholder profile, and the Applied Fraud Policy to determine the authorization approval through interbanking. In this manner it is the security profile and preferences of the individual account holder which determine the fraud detection, based on the transaction behavior and customer profile. We root the issues and proposed semantic mediation in component based approach as once enterprise lose its visibility of semantic interoperability, ad-hoc enhancement cause recursive issue and risk expands rapidly.

Componentization and semantics mediation are mandatory in maintaining a coherence of service based industry. Component is actually the object of semantic mediation. Initially, component validation and analysis consists of steps in validation of business components, service components, and physical components, common component parsing analysis, and service aggregation parsing analysis. Our architecture interoperability is based on component validation, component design, micro process mediation and dimensional services. The realization of this top-down vertical integration approach requires robust mediation to support the inter-communication between component-component and component-repository. In other words, the concept needs a macro framework and physical support in micro integration service bus such as the middleware in realization of run-time services. However, semantic mediation is the foundation as each component need to transmit particular message in maintaining the ontology of service interoperability. The weakness in semantic mediation has become one of the top issues in this decade as business strategy finds itself losing to its competition in service transformation. We recognize that this issue has been created by the neglect of semantic strategy; a confused design has thus resulted in businesses losing their enterprise governance for the accommodation of visibility in semantics, syntax and architecture alignment. In other words, enterprise has been unable to effectively leverage business, service and technology into a common vision. Semantic mediation in enterprise integration requires following technologies to achieve its goal: metadata, semantics, component, interoperability, common service and service aggregation. Each area can be a major factor and require support from other sub areas. Since our concept is still a pilot in this field and its exploration has already generated interest within the industry, we are currently moving our concept into realization; meanwhile, we would like to further clarify our motivation as follows:

Our main focus is not limited to the subject area; rather our concerns are in how these technologies work together. From our literature review, we found that the direction of enterprise integration urgently needs a new direction as most of the literature limits its coverage to the subject area of point solution; very few covered the architecture level in how to construct a "big picture" based on the knowledge we have. On the other hand, from the limited literature which already sheds light on architecture, we found that the concepts posed did not explore industrial innovation and real demands. Our final goal is service virtualization; this concept indicates that in the virtualization stage enterprise integration reaches its highest maturity and seamless integration across enterprise and industry. We believe this objective is one of the key tasks in this century for most organizations; industry requires a common objective and roadmap to achieve this goal, thus researchers working from the subject of point solution can complement each other in creating a homogeneous environment. Our integration approaches are based on the metadata component which is built on top of the foundation of semantic interoperability. This means our method is not based on peer-to-peer, system or application integration, rather our interest is in following the nature of business. This interest includes how the

Copyright © 2007 Inderscience Enterprises Ltd.

business is composed and what strategy synthesizes the business and technology aspects into two directions made up of architecture strategy and the project life cycle. We observe and use a working technology to construct the strategic "framework" and "mechanism" as the foundation of service interoperability (Wu, 2007). We believe the nature of business and human behavior need to be taken into account as these can never be replaced by merely using technical assets.

Literature review

It has been recognized that IT industry in financial sector is one of the most highly complex among the various sectors, and along with the complexity comes high risk. To eliminate the risk, our assumption is: The higher occupancy of common services within the enterprise, the less suffering will be caused in each IT and business transformation, as technology and business innovation and transformation, while creating opportunity, may impact the enterprise seriously if the transformation foundation is not solid. Weakness in enterprise skeleton and in interoperability supported by common components, may cause high risk in each transformation necessary to survival in the next stage. To cope with these issues, we followed the enterprise integration strategy in our recent research which includes: validation of business and service components, service analysis and transformation, service design, information aggregation and service provisioning. Our research in enterprise integration in addressing these issues has been divided into four areas: validation, analysis, design and virtualization (Lytras, 2006).

Our semantics strategy is focused on architecture interoperability by using semantic mediation. There are two areas of semantics that need to be clarified; the first one is business semantics which is strategic semantics or what the business wants to do. The second area is semantics content i.e. the information needed in responding to the service request to achieve a particular business semantic (Moulton et al, 2003). Our focus as regards interoperability is how to mediate these two areas of semantics in achieving a business goal by a particular dimensional service.

According to P. Kruchten, a component is a non-trivial, nearly independent and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture. Desmond DSouza defines vertical integration in his "Model-Driven Architecture and Integration Opportunities and Challenges," stating that vertical integration comes from different levels of abstraction of the same system, from physical data models, logical data models, networks, applications specifications, component assemblies, business process models and business goals and strategies (DSouza, 2001).

Common interface is also a means to achieve common components. S. Melnik proposes a universal interface which avoids common models and languages; this is the Generic Interoperability Framework which was developed to facilitate the integration of heterogeneous information systems. Within this framework individual communication protocols, data manipulation languages and data are represented in a generic manner while preserving their ontological variety; this process indicates how a mediation infrastructure can be built using canonical wrappers, operational requirements of integration and layered architecture for component design (Melnik, 1999). D. Oberle, S. Staab and A. Eberhart, in their Kaon server design, show several potential sources of semantic metadata. These sources provide input for the framework regarding the semantic management of software components. This includes deployment descriptors, Web and application server configuration files, source code information or annotations and database metadata. The component metadata collector parses this information and automatically integrates it into the ontology as semantic metadata (instances). The inference engine is embedded in the application server and applies the ontology and semantic metadata for reasoning and querying. The application server's core functionality and the administration console both use the reasoning capability. The latter allows the developer to browse and query the ontology at runtime so that configurations may be queried or checked to avoid inconsistent system configurations during development and at runtime (Oberle et al, 2006).

J. Qin and J. Prado, in their "The Semantic and Syntactic Model of Metadata," further explain the

Copyright © 2007 Inderscience Enterprises Ltd.

ontology-based approach for taking both semantics and syntax into modular model metadata, particularly in light of gaps between metadata semantics and syntactic structures resulting in duplicate efforts in binding the same data elements with various languages and application programs. This has led to widely varied data binding models and implementations, while common semantic elements in metadata schemas such as title, author, subject index terms, and description are often far from enough when finer metadata representation is needed (Qin et al. 2006), (Qin et al. 2004). This confusion forces developers to expand the metadata semantics with methods and technology suitable in their specific context, which results in widely varied practices and duplicate efforts. We feel this concept explains our approach and that in the real world the co-existence of semantic and syntax is not necessary. As this is technically not possible, instead of dealing with fragmentation from scrambled elements we instead proposed component-based semantic mediation in which each component is a "qualified" atomic element which can be executed solely or proceeds to dimensional provisioning combined with other components via component interactivities. However, there are some rules that we need in order to prevent in these interactivities what is called "problematic architecture" (Davis, 2001). In our approach, we do not rely on syntax rule in semantic mediation because when we build up the physical component we have already identified each component with specification in the object, action to be taken, interfacing and event details, so the component has enough capability to work independently or "co-work" with others. Our concerns are the metadata the component transmits, the governance the component mediates, and the service the component interoperates with; these will be addressed in this paper with a proposed resolution. We would first like to put some emphasis on the meaning of the achievement of integration and interoperability by semantic mediation.

Semantic interoperability attempts to homogenize disparities through standards. The interest of many researchers regarding semantic interoperability is how people and social groups with different perspectives identify and possibly resolve their semantic differences (Harvey, 1999). As far as we know, the construction of information communities in heterogeneous institutional and disciplinary environments calls for frameworks to conceptualize and articulate these semantic differences. A reason for this is that a geographical heterogeneous state in semantics forces peers to follow a power law distribution which emerges frequently in networked interactions (Cambrin, 2004) (Aberer et al, 2002). The other concern is that implicit information embedded in semantic web graphs, such as topography, clusters and disconnected sub graphs, is difficult to extract from text files (Mutton, 2003). Therefore, we would like to propose the service transformation approaches, either bottom up or top down, to guide the framework implementation which has already been launched in several software life cycles and project governance. There have been many attempts at defining the concept of interoperability. A few examples are given following: Interoperability is the ability of multiple systems with different hardware and software platforms, data structures, and interfaces to exchange data with minimal loss of content and functionality (NISO, 2004). It is the ability of two or more systems or components to exchange information and use the exchanged information without special effort on either system (CCDA, 2000). It is also the compatibility of two or more systems in order to exchange information and data and use the exchanged information and data without any special manipulation (Taylor, 2004).

In enterprise integration, our view of semantic interoperability is this it is involved in the degree of flexibility two different metadata sets have in working together and understanding each other using a common terminology. Apart from this peer-to-peer concept, semantic interoperability has special meaning in our research, indicating the strategy and capability of cross layer transformation which takes business semantics into component-based implementation. Thus the atomic element, in the form of a service component, "inherits" the micro semantic and syntax in mediating the information element which is in metadata form. Semantic interoperability within an enterprise is possible from our approach, as enterprise architecture is capable of leveraging all system, business domain, terminology, organization, database and process into a metadata repository. Once this enterprise

Copyright © 2007 Inderscience Enterprises Ltd.

repository is constructed, the enterprise gains governance, reuse, standardization and transparency which are the foundations of semantic interoperability at the enterprise level (Panfilis, 2001).

Semantic interoperability in industry is much more complicated than within enterprise architecture. The first barrier is the use of different metadata schemas. To circumvent the problem, various mappings or crosswalks have been devised to translate between metadata systems. An example of this is d2m, which converts Dublin Core to a selection of Scandinavian MARC formats. Research is also underway focusing on the use of XSL style sheets to transform XML metadata records from one standard to another (Pierre et al, 1998). The National Institute of Standards and Technology (NIST) estimates that the lack of interoperability costs the U.S. capital facilities industry alone $15.8 billion per year, representing 1-2% of industry revenue; this estimate is likely to be only a portion of the total cost of inadequate interoperability (Gallaher et al., 2004). It is the weakness in interoperability and the consequent risks to industry, defense and government that are the motivation for our research. Therefore, before we investigate semantic interoperability, we need to understand where the problem came from. We subsequently performed three emulation experiments on semantic mediation to aid in our understanding of the behaviors of semantic metadata, service mediation and integration.

Case study – Semantics and componentization in credit card transaction

For further understanding of how semantic mediation and componentization interacts in financial industry, we break down the business component into the following use-case scenarios for further development. The business scenario is related to credit card transaction between cardholder, merchant and interbanking. We deconstruct the business components into the following service use cases in sequential order with business scenarios shown in Figure 1:

- ✧use case-1: Buyer sends payment information to the merchant server.
- ✧use case-2: Merchant takes payment information from the cardholder and sends to acquirer.
- ✧use case-3: Acquirer sends an authorization request to the issuer over the interbank network.
- ✧use case-4: Acquirer notifies merchant about the status of authorization.
- ✧use case-5: Merchant presents the charge to the acquiring bank.
- ✧use case-6: Acquirer sends a settlement request to the issuer.
- ✧use case-7: Issuer charges the buyer's credit card account.
- ✧use case-8: Buyer pays the charges to the bank.

In our current research, we are focusing on the interoperability of service provisioning and virtualization. In the conventional credit card service, the interbanking doesn't maintain a "fraud detection policy" service at the account level. Mostly this shortage is the result of conventional silo implementation and limitation of multi-site interconnectivity such that a daily program is used to update the hard-coded matrix within the whole authorization process.

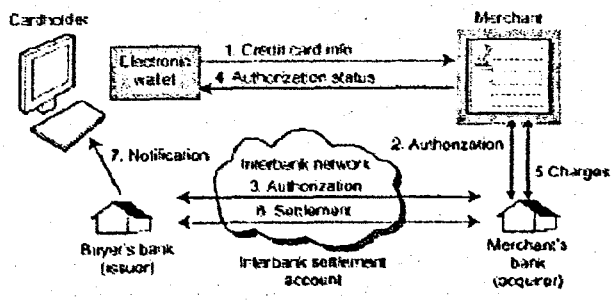


Figure 1. Business use case in the financial industry

The issue is with a non-component-based environment in which implementation can cause high impact on the enterprise, and may even be unachievable between different organizations having a common metadata to handle. In a non-component-based environment, fraud detection apply generic

Copyright © 2007 Inderscience Enterprises Ltd.

rules and policy to screen each transaction, the detection between transaction, policy, account and branch doesn't maintain a component base semantics and contextual coherence in the detection, it result in many transactions are blocked as "suspect" or "reject" due to the enterprise doesn't have fine-grained component to mediate the semantics and maps with the policy in a dynamic way, despite the enterprise has enough metadata to manage the policy and account base computing, the policy detection frequently rollback the transactions, which are not be fraud case, due to lack of a component base dynamic semantics exchange problems. We illustrates the add-on component provisioning a new service in fraud detection as a dynamic option based on customer profile, as a supplementary solution

Semantic component implementation in this case study involved in: dynamically retrieve cardholder profile and the applied fraud policy to determine the authorization approval through interbanking, so each account holder's security profile and preference for the card issuer or dealers will determine the individual fraud detection based on the transaction behavior and customer profile. Service components derived from the business use case listed previously, are the mediation between business and technical elements which integrate all business semantics, interfaces and technical configurations into a logical component. Initially all the service components are at an atomic level (Staab et al, 2001) (Valkeapää et al, 2006).

Part I. Component design

Prior to the component design, the business component selection, deconstruction for service component validation and analysis are required in component-based design for enterprise integration objective. We will not detail the process as these areas were the focus of our previous research. We introduced a group decision model in the validation process, so business components with their associated service components are selected and validated through the decision process.

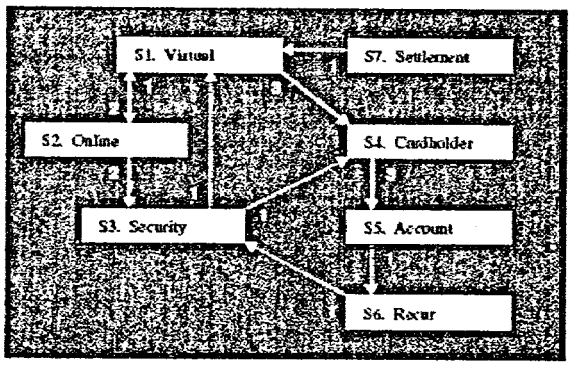


Figure 2. Referential integrity of component interconnectivity

Table 1 shows the alignment of the validated business component, the deconstructed use case, the service component and component elements. We further detailed the rational model in analyzing common components and aggregated services by using a use case parser to determine enterprise components in forming the skeleton of integration. Aggregation is a kind of service normalization in grouping all atomic service elements from fine-grained to coarse-grained components.

We also introduced the component design and interconnectivity for how the components communicate with each other and how the micro process of the business semantic is derived in each component. Figure2 further shows the referential integrity of component interconnectivity for how the components relate to each other.

The component element is the physical element that mediates service and information. The design of service components follows our previous strategy, by using business semantic injection to take the context into the semantic and micro processes, so that each component has its contract-level responsibility in provisioning a subset of services, and the interface helps the coordination between components and metadata build the foundation of interoperability within the service layer and between layers. In the following paragraph, we take the results from component design into the service level, and meanwhile we apply automata theory to build up the rule-based service aggregation at the enterprise level. As we are still at a pioneer stage in applying automata theory in service

Copyright © 2007 Inderscience Enterprises Ltd.

provisioning and aggregation, we understand this cannot be accomplished within a short period, and further research efforts will be undertaken following the current work (Barcia et al, 2005).

In any component context a formal interface contract for the invocation syntax is important. The semantics issue (the meaning of parameters and so forth) has to be solved as well (domain modeling). This is crucial to any business-to-business (B2B) and dynamic invocation scenario. Such scenarios are cornerstones of the service component vision of being flexible and agile in response to the new business needs in a world of mergers and acquisitions, business transformation, globalization and other relevant factors (Zimmermann, 2004).

The string $\Phi(r_n, r_m) = \Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m$ is a path from r_n to r_m, we call the "service route" and

$r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$ a "composite service" between r_{i-1} and r_{i+1}, we use the symbol $\Phi_{i, i+1}$ to denote a composite service.

When the business architect checks the result from semantic event aggregator - ε^m which is more an upper-layer use case scenario, service route - $\Phi_{i, i+1}$ which is in the service layer and its frequency - $F(\Phi_{i, i+1})$, if $F(\Phi_{i, i+1})$ is much higher than threshold σ, then from the business perspective the event aggregator ε^m either consists of many composite services or is business critical. This means the aggregation service is the key within business and is crucial in daily operation from service perspectives, and that fine tuning is feasible from a multi-fine-grained boundary to a coarse-grained service. We call this the "main route" of the enterprise skeleton; in other words this is the highway between A and B. Virtual coarse-grained service eliminates the boundary and leverages many fine-grained components which can be easily configured by adding or modifying the component dynamically to achieve the broader coverage of main route services. For this reason main route is one of the crucial services in enterprise integration. One example is open service account, an open billing account and provisioning options for a new customer.

This constitutes a main route for "new customer service" coarse-grained service, though it is aligned with fine-grained components to support customer preference, service criteria and billing options which are branches within this main route.

The importance of determining the common component and service aggregation is in building up an enterprise skeleton of vertical interoperability between business, service and information.

This task is crucial, as in the business domain analysis stage, the impact area and scope to business components are minimized in decision making. Additionally, the business component is deconstructed into service components and physical elements for top-down alignment while the common component is not simply a symbolic service but instead, it requires a bottom-up approach to work on common areas; this serves to fine tune the common components and interfaces from the perspective of enterprise integrity and performance.

Figure 3 shows how common components and composite component service aggregation form the skeleton of enterprise integration; the skeleton extends its service to the derived service and forms the interoperability with upward linkage to business and downward to physical components. The service aggregation is also important as the top-down deconstruction followed by bottom-up composition forms the skeleton of vertical interoperability and potential component-based implementation of the service process.

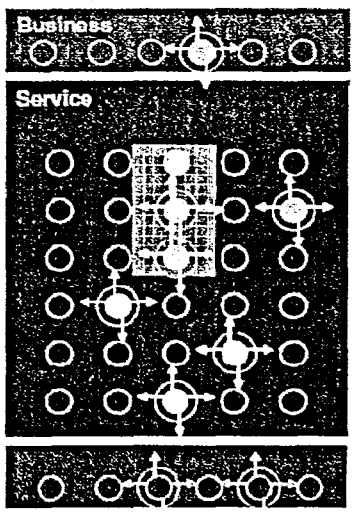


Figure 3. Transformation of common components and service aggregation

Copyright © 2007 Inderscience Enterprises Ltd.

Part II. Semantic mediation

When the service event Φi (i <>1) is read and triggers the service component, we express the state transformation as follows:

$\varepsilon^{i}: r_{i-1} \times \Phi_i \rightarrow r_i$, we further derive the transformation as follows:

$\varepsilon^{i+1} = (r_i) \times \Phi_{i+1} \rightarrow r_{i+1} = (r_{i-1} \times \Phi_i) \times \Phi_{i+1} \rightarrow r_{i+1}$ which is in equilibrium to

$r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$, and derives

$\varepsilon^{nm}: r_n \times (\Phi_{n+1} \times \Phi_{n+2} \ldots\ldots \times \Phi_m) \rightarrow r_{m\ (n<m)}$, we call this the "semantic event aggregator."

This means, the event 'aggregate' service component by using semantic metadata. When we eliminate number of event, we already simplify and build a potential main route for a particular service and bypass intermediate steps by using components' interconnectivity and service plan to transform service into a solid enterprise skeleton. In any component context a formal interface contract for the invocation syntax is important. The semantics issue (the meaning of parameters and so forth) has to be solved as well (domain modeling). This is crucial to any business-to-business (B2B) and dynamic invocation scenario. Such scenarios are cornerstones of the service component vision of being flexible and agile in response to the new business needs in a world of mergers and acquisitions, business transformation, globalization and other relevant factors.

The string $\Phi(r_n, r_m) = \Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m$ is a path from r_n to r_m, we call "service route" and $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$ a "composite service" between r_{i-1} and r_{i+1}, we use symbol $\Phi_{i,\ i+1}$ to denote composite service. When frequency $F(\Phi_{i,\ i+1}) = F(\Phi_i \times \Phi_{i+1})$ is higher than threshold σ, we aggregate the concatenated alphabet (part of the string) $\Phi_i \times \Phi_{i+1}$ into $\Phi_{i,\ i+1}$, this implies the original state r_i between r_{i-1} and r_{i+1} will no longer exist for this composite service.

From the information metadata perspective, to achieve the service provisioning in single view of the customer in credit card industry, we need to aggregate all fragmented data sources into metadata and business intelligence to build the service virtualization foundation based on the integrity between business, service and information. M. Schmalz argued that database schema and application code can be efficiently derived from various types of schema representations, particularly the relational model, and supports comparison of a wide variety of schema and code constructs (Schmalz, 2003). Once we create insight into the customer's preference and transaction behavior, we can effectively add high-value services on top of the information metadata, which represent the accuracy of customer, summary information and business intelligence; this is the information aggregation metadata concept.

The transformation therefore requires an event to incur the service with the intelligence to either predict the next service or to verify that the called service is recognized. We use the state concept discussed above to mediate the business, service event and information in the "time" dimension. The state of the service layer and the business-service-information coherence are changed once we trigger an event and the service is successfully executed. The alphabet read in automata is leveraged to service component trigger and execution while the state change is leveraged to transformation within the service layer. At the very beginning of a use case, the first event triggers service Φ_1

$\varepsilon^{1}: r_0 \times \Phi_1 \rightarrow r_1$

When the service event Φi (i <>1) is read and triggers the service component, we express the state transformation as follows:

$\varepsilon^{i}: r_{i-1} \times \Phi_i \rightarrow r_i$ and we further derive the transformation as follows:

$\varepsilon^{i+1} = (r_i) \times \Phi_{i+1} \rightarrow r_{i+1} = (r_{i-1} \times \Phi_i) \times \Phi_{i+1} \rightarrow r_{i+1}$

which is in equilibrium to $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$, and derives

$\varepsilon^{nm}: r_n \times (\Phi_{n+1} \times \Phi_{n+2} \ldots\ldots \times \Phi_m) \rightarrow r_{m\ (n<m)}$ We call this the "semantic event aggregator."

As the final step of a use case, the final event trigger service Φ_n

$\varepsilon^{n}: r_{n-1} \times \Phi_n \rightarrow r_n$, Φ_n is a terminator, an example of which is the settlement of a daily transaction.

Copyright © 2007 Inderscience Enterprises Ltd.

To investigate the fundamental mechanism for the further understanding of the meaning of semantic mediation, we execute benchmarking by using experiments to emulate semantic mediation. Our objective is to demonstrate how to cope with versatile business change, and which pattern is a better practice for semantic mediation. Following are the scenario and result from three experiments:

Experiment I - We use an SQL-based computing model to emulate the phenomenon of semantic interoperability. The computing model consists of the parser, the optimizer, and the row source generator which form the SQL compiler. Compiling the SQL statements into a shared cursor, and associated with the shared cursor, is the execution plan. The parser performs two functions: SQL analysis which checks SQL statements for correct syntax, and semantic analysis which checks, for example, to be certain that the current database objects and object attributes referenced are correctly mapped. As our interest is not in SQL analysis, we are instead very focused on semantic analysis.

The row source generator receives the optimal plan from the optimizer. It outputs the execution plan for the SQL statement. The execution plan is a collection of row sources structured in the form of a tree which determine whether semantic mediation is applied and to what degree it will be utilized. A row source is an iterative control structure that processes a set of rows, one row at a time, in an iterated manner.

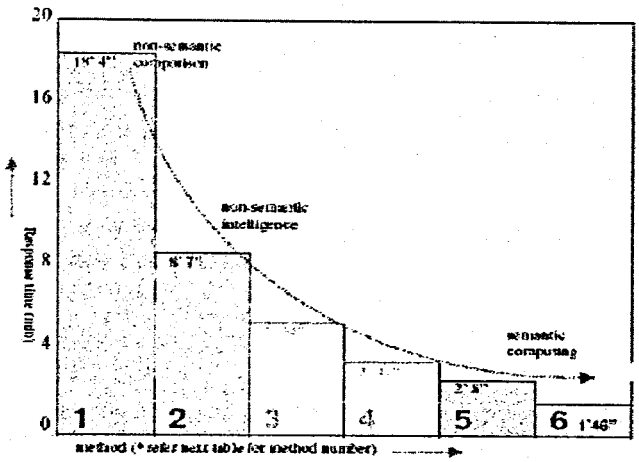


Figure 4 Experiment result from different semantic strategy

A row source produces a row set and the performance, reflected from response time, is an index of the performance of semantic mediation. With the same query, we can see that when it is interpreted in different ways and with or without semantic mediation, the results are all different. The response time dropped from 18' 4" to 8' 7", 2' 8", 1'46" for more than a ten times improvement from conventional SQL to full cursor computing. The benchmarking result was shown in Figure 4 with an explanation for how semantic strategy was applied in each scenario. The sql code and result are shown in appendix 1 & 2.

Experiment II - In our second experiment, we used a similar model to experiment I to emulate the scenarios of random query and semantic cursor computing, and produced an execution plan as well as a physical read and response time. From our experience, the response time reflects how easy the SQL query mediates the semantic metadata and how the physical read drops from 166,386 to 5,594 (3.5%). This means that in a non-semantic world, 96.5% miss from a random read; systems suffer 96.5% resource abuse; and users waste 22 times in response time (1'10" vs. 3"). Figure 5 gives symbolic illustration in differentiating hash computing and semantic computing.

Experiment III - For the scenario we used in experiment III, the source site is the Unix customer management system and the remote sites are the customer care center (MS SQL-server case) and call center (Oracle/Unix case); both cases can be expanded to an unlimited or heterogeneous database, a different operating system and different data format. It guarantees a single source of truth and single point data manipulation. The front site

Copyright © 2007 Inderscience Enterprises Ltd.

in this experiment is the VB program; the MS-server front end takes the data entry from customer manipulation for the program of promotion and rating, automatically generating scripts and sets of data formats to transmit to remote for execution.

More precisely, the engine we built takes care of the schema mapping (between source and remote) to transform the data to remote DB format and generate the command files to transmit the SQL/shell script to the remote site by file transfer and then to execute it remotely. The whole process takes no extra effort at remote sites and the whole process takes only 5 seconds for several million of records to synchronize data for each site. Following are some simple examples used in our experiment III in demonstrating semantic mediation across systems, which support part of our assumptions. Based on the experiment results, we produce following assumption:

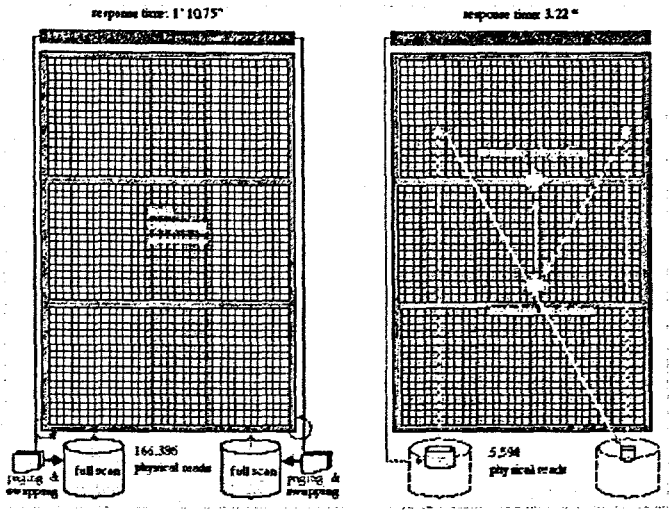


Figure 5 Symbolic illustration between hash and semantic computing

Assumption 1 – Scenarios for emulation

As shown in Figure 6, we leverage and use a typical computing model (application, memory and database) to emulate the service architecture model (business, service and data) as we found the mechanism of semantic cache memory computing can be leveraged to a flexible service layer in mediating semantic between business and data.

So the memory computing in the conventional, intelligence and semantic patterns model are all leveraged to their associated service patterns model.

As mentioned above, a service pattern is the implementation and realization of dimensional design that consists of the principles and methods for how components interact with each other; they can be classified into the following four patterns:

- Pattern 1: Coarse-grained "glued" type: In a conventional peer-to-peer integration,service mediation is only limited to the block level; most of the components are large block and are "glued together."
- Pattern 2: Fine-grained "glued"

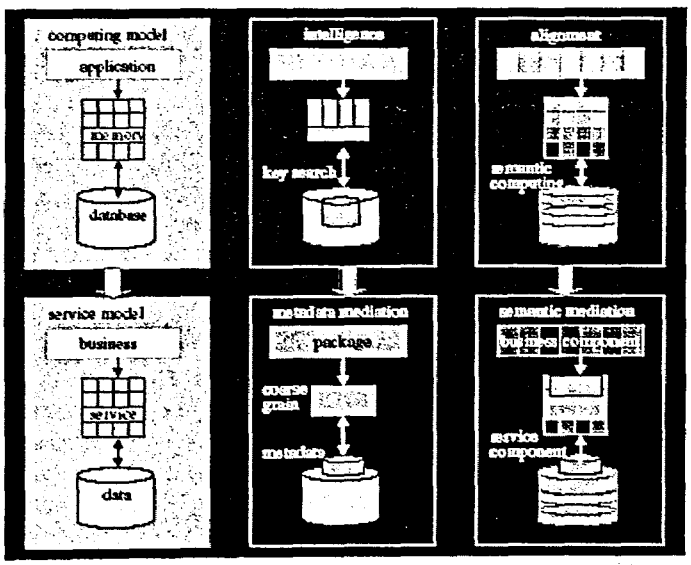


Figure 6 Leverage between computing model and service model from semantic perspective

Copyright © 2007 Inderscience Enterprises Ltd.

type: In a non-semantic component-based integration, we produce fine-grained interoperability between components in the construction of ad hoc type mediation; in this way the whole enterprise of different systems and applications can build their interoperability in a horizontal direction without a clear picture of the vertical direction.

- Pattern 3: Thin layer service type: In a partial semantic mediation integration, which we call the thin layer service, it is possible to construct the visibility of a virtual service across enterprise; in this way part of the component has the opportunity and can start to form upward aggregation into the service layer, which is the virtual concatenation output from component.
- Pattern 4: Thick layer service type: In this full semantic mediation integration, the thick layer of service provides high agility and virtualization of the services layer in constructing a dynamic and enriched service.

Assumption 2 – Emulation method

Given each computing model facilitated by the same utilities, cache memory utilization is critical in performance and semantic computing is the key of cache memory efficiency. As memory provides mediation of semantic content and data in metadata alignment, for this reason we leverage cache memory to metadata mediation. All the emulations are based on the assumption that a universal semantic, schema and syntax are achievable; in this way we achieve common practices in federation governance. The methodology we proposed in maintaining this common practice includes:

- Micro design of component, which takes semantic and syntax into the component level process
- Mediation of semantic within service component in provisioning upward and downward service
- Service interoperability in enterprise and industry in achieving service virtualization

The key component in this mediation is called the context mediator, the function of which is to identify and construct the semantic values being sent in order to determine when the exchange is meaningful as well as to convert the semantic values to the form required by the receiver (Sciore, 1994). Our observation in the experiment produced the evidence that a well-designed semantic interoperability within the computing model reflects a better performance of data query. We are seeking the proof of the following statement: "Better cache memory utilization provides for better semantic mediation; the scenario of mediating between database and application can be leveraged to service mediation between information and business."

Assumption 3 – Macro view mediation in a heterogeneous environment

In our experiment III, we tried to apply our architecture innovation ESI-SRE (Enterprise Semantic Interoperability and Self Reproducing Engine) to synchronize the enterprise master data change to remote sites by using metadata broadcasting. We wish to demonstrate the following: 1) our experiment shows that semantic interoperability within a service layer or between systems can be fully transparent, thus environment is not a barrier or concern in semantic mediation. 2) master data centric and pre-built service - Single point data manipulation is necessary to maintain an enterprise consistency of the master data; this is the pre-built alignment that referential integrity extends from the service to enterprise service mediation in data synchronization. 3) both DDL and DML can be self reproducing depending on the system, language and semantic requirements; although semantic mediation can be pre-built, many of them need to be self-produced dynamically. This is why our ESI-SRE solution supports the dynamic self-producing case, so that the engine looks into the environment for parameter fill-in and generates executable codes for further processing. 4) strategic semantic mediation - A central control of semantic mediation in maintaining metadata repository, information consistency and operational synchronization is mandatory in semantic mediation. This is the key to what we demonstrate in the experiment. 5) physical infrastructure mediation – In our component-based semantic mediation, the data, system function, executable code and API are all components associated with a service. They are loosely-coupled and transportable in a homogeneous environment facilitated by service transparency. Examples include our ability to "ftp" (file transfer) a self-

Copyright © 2007 Inderscience Enterprises Ltd.

generated code as a component to remote sites, or mediate associated data records to a targeted site, or provision a variety of services by virtual system topology, mediation redundancy, as well as transmission balancing and mediating broadcasting. These are all critical and achievable.

When we reviewed the results of our experiments in relation to the behavior of these patterns by using conventional parsing and a full-table scan in swapping and paging, we found all to be similar to pattern 1. The scenario of implementation point parsing was used through the whole program, as any minor change may cause a high impact. Patterns 3 and 4 are leveraged to semantic mediation. By using metadata mediation and semantic integration through cache memory computing, the physical read drops to 3.3% and the response time was cut to 4.3% when using conventional hash comparing and joining. Semantic mediation is the key in this enhancement for the following reason:

Hash computing consumes cache memory by using a full-table scan and swapping in/out the full tables. The huge volume table access, storage I/O and swap in/out cause inefficient computing, as element read-in can cause mass changes in a table scan. The 33 times physical read of hash comparing rather than semantic mediation makes this evident. This is mostly caused by the full-table scan due to weak semantic strategy; the high-volume data takes whole memory and recursively retrieves the database by swapping in/out. Comparatively, semantic computing uses cache memory in cataloging and indexing to focus on the mediator, thus the scope becomes small and the semantics are fully utilized in the mediation between "mediator" and data. By taking this scenario to semantic service implementation, we derived the following equation: $\Delta P = \Gamma(\Delta\Phi) = \Gamma(\Sigma\ \Phi_n(C_p) - \Sigma\ \Phi_{n-1}(C_q)) \; ! = \Gamma(\Sigma\ \Phi_n(C_p)) - \Gamma(\Sigma\ \Phi_{n-1}(C_q))$
We therefore, benchmarked the above service pattern as followed:

Pattern 1 has nothing to do with effective semantic techniques within enterprise; the consolidation of components is limited only to system interfacing such as API (application protocol interface), database link or file transfer, and all implementations are in ad hoc form, therefore no sharing or reuse can be expected in the entire project life cycle.

Pattern 2 is the first step of componentization. All non-function components such as the data model and system utilities are decomposed into a loosely coupled environment. Although the business functions are still ambiguous, some service dimensions form a rough outline in the strategic level.

Pattern 3 is the initial step of service mediation; a thin layer of the service component takes over and represents the physical component in semantic mediation, expanding the mediation in a business direction. Micro interoperability, once identified and implemented, has the capacity to explore the common component and service in contributing to the enterprise skeleton. In this stage, service dimension and transformation start to replace the conventional design. As common service becomes mature in this stage, components continue to aggregate and explore their upward service route.

Pattern 4 is a full semantic mediation provided by a thick service layer. As our focus shifts from component-based techniques to semantic mediation and virtual alignment, the mutuality of micro mediation within the component, together with macro interoperability between the business and component, clearly provide a versatile service in identifying either existing services or new services. An enriched services layer produces coarse-grained services which are aggregated from fine-grained components. Service dimensions, therefore, become an easily configured soft area for the provision of a full range of services. The notable strategies and characteristics of interoperability include: a universal schema and metadata in the information layer; semantic mediation between business, service and component; interoperability between three repositories; and business decomposition in the upper layer. Among these, the determination of the service atomic component is the key for moving component-based architecture into semantic service. We are unable to merely rely on the configuration of the component in achieving dimensional service, as it does not make too much sense

Copyright © 2007 Inderscience Enterprises Ltd.

in semantic mediation within enterprise integration. However, we certainly need to apply component-centric design to fulfill interoperability requirements. The reasons for this are: 1) The service atomic component is the mediation unit between business and IT as this atomic element loosely binds business and IT terminology in a common view; only through this single point design do we have the opportunity to align enterprise architecture into interoperability; 2) From this single point design we can validate associated metadata in the repository for their analysis in noun, verb and attribute as an atomic element while using business semantic terms to mediate pre-built component sets which have been configured for that service; 3) The service component can be easily built, revised or aggregated to achieve any service change caused by business change because most of the physical component design is an atomic level function. The mechanism of interfacing and parameter transfer are all keys to how each component performs; therefore, the mechanism of the "integrated functions" should not be vaporized, rather they need to be stable and durable. This means the virtual service which is mounted on the supporting components should effectively apply metadata in the repository to mediate other services in provisioning a service dimension; once this service is validated, the metadata which holds the mediation information should be stored in a knowledge-based repository for reuse or revision; 4) The service layer is an easily configurable area which we can leverage to the scenarios of cache memory computing in mediating application and database storage. The syntax, logic, semantic and execution plans such as security features are all under metadata control; these are stored in repository and ready for configuration.

Interference of component and semantics in financial industry

The micro process in component design follows the top-down direction and takes the deconstruction output to derive the service component. We use the interbanking example of the credit card industry to demonstrate the transaction between cardholder (payer), merchant (payee), bank (transaction handler) and credit card authority (interbanking). We deconstruct the business component of credit card transaction into several use cases, as follows. 1) Consumer contacts merchant to encrypt and decrypt security keys, and to request certificates. 2) Payment is transferred from merchant to acquirer (the merchant's bank), and then to issuer through interbank network to authenticate the payment information. 3) Acquirer notifies merchant about the status for merchant to fill the order. 4) Merchant requests acquiring bank regarding settlement. 5) Issuer charges the consumer's credit card and notifies cardholder. 6) Consumer pays fee, and acquirer then credits merchant for the exact fee.

The conventional design doesn't have the dynamic control to have customer insight while the issues are resolved in component-based design. This is the service transformation and provisioning example which mediates the enterprise components in forming the skeleton. Service components derived from the business use case are the mediation between business and technical elements which integrate all business semantics, interfaces and technical configurations into a logical component; initially all the service components are at the atomic level (Staab, 2001) (Valkeapää, 2006). In our component validation and analysis research, we applied a group decision model in the validation process, so the business component with its associated service components are selected and validated through the decision process. We further detailed the rational model in analyzing common components and aggregated services by using a use case parser to determine enterprise components in forming the skeleton of integration. Aggregation is a kind of service normalization in grouping all atomic service elements from fine-grained to coarse-grained components.

We also introduced the component design and interconnectivity for how the components communicate with each other and how the micro process of business semantics is derived in each component. The component element is the physical element that mediates service and information; the design of service components follows our previous strategy, by using business semantic injection to take the context into the semantic and micro processes, so that each component has its contract level responsibility in provisioning a subset of services, and the interface helps the coordination between components and metadata build the foundation of interoperability within the service layer and between layers.

Copyright © 2007 Inderscience Enterprises Ltd.

Conclusion

The financial industry is facing a big challenge within enterprise integration as the service concept is moving, global policy and regulation is changing and the high demands in transforming the system silos to an industry and global standard is not an easy task. We gave the example of a typical issue in a credit card scenario for how to narrow down the root issue within a small scope as a service component while maintaining its alignment to both business and technical components; this service component architecture speeds up the change and dramatically enhances the integrity and responsiveness. An example of service provisioning would be the case where several key industry vendors have built up an industry framework for service providers in that particular industry to follow, starting from business domain analysis and working through to component deployment. So each service provider benefits from the industry common services and common interface, particularly those B2B, B2C, merger and acquire cases. Any business activity becomes easier when the various parties involved have a common understanding and consensus on business terms, service definitions, interfaces and component specification, and these become transparent and common practices (Zimmermann, 2006).

The enterprise integration life cycle consists of validation, analysis, design and virtualization. Enterprise interoperability is built on the foundation of component-based design; we applied both macro and micro process in constructing the foundation, and we further aggregated the components into common components and aggregation services. This encompasses component-based design and interconnectivity, transformation of service aggregation through rule-based design, service provisioning in a particular dimension and service virtualization. In component design we introduced the micro mechanisms within components as well as component interconnectivity. We classified component design into interface, implementation and metadata areas to guide the design from the internal to the external. This will be introduced in a separate research paper. The important concept we derived in this research is that once the semantics of business, service and information constructs an industry standard, the organization's vertical skeleton has the opportunity to integrate across multiple organizations for an easier global practice; this is a long-term industry objective which we call service virtualization, and in this pioneer work we are influencing the financial industry in achieving the objective of component-based integration and semantic mediation.

Our observation of semantic behavior supported by emulation experiments shows that the combination of metadata and service mediation are the keys in dimensional services. Metadata is the strategy of mediation content, while service mediation is the mechanism to interoperate components in a particular service direction which we call dimensional service. Our observation of semantic behavior supported by emulation experiments shows that the combination of metadata and service mediation are the keys in dimensional services. Metadata is the strategy of mediation content, while service mediation is the mechanism to interoperate components in a particular service direction which we call dimensional service.

To further understand the service mediation mechanism, we leveraged the semantic mediation from the experiments on cache memory computing to service mediation; from our experiments we learned that service mediation requires both intelligence and knowledge-based semantic metadata for micro processes design.

Additionally, the experiments indicate that a dramatic improvement is achievable by using semantic metadata in cache computing; this shows the importance of semantic mediation within component-based enterprise integration. We also learned from the emulation experiments that service mediation requires a micro process alignment of metadata in semantic grouping, mapping and intelligence in self-generating automation. We classify service design into four basic patterns: 1) the conventional design; 2) the component-based without intelligent alignment; 3) the aligned component-based consisting of a thin service layer; and 4) an aligned thick layer. Our modeling analysis shows that a

Copyright © 2007 Inderscience Enterprises Ltd.

thick service layer aligned by using the mixed coarse-grained and fine-grained option is consistently the best practice. The conventional design in non-component based, which lacks a semantic mediator between business and technology, should be transformed accordingly. We further explored the application of AHP pair-wise comparison and eigenvalue techniques in deriving the relative weight of a component toward a dimensional service; we found that this aids our component evaluation and produces a service map prior to provisioning a dimensional service.

Copyright © 2007 Inderscience Enterprises Ltd.

References

Curtis, K., Foster, P. W., Stentiford, F., (1999), "Metadata - The Key to Content Management Services", Proceedings of the Meta-Data '99, The third IEEE Meta-Data conference, April 6-7, 1999, Maryland, USA

Wu, R., (2007), "Enterprise Vertical Integration in Metadata Strategy", The IBFR- GCBF Proceedings – Vol 2, No 1 ISSN 1931-0285, Texas USA January 2007

M. Lytras, (2006), "The Semantic Electronic Government: knowledge management for citizen relationship and new assessment scenarios", Electronic Government, an International Journal 2006 - Vol. 3, No.1 pp. 5 – 17

Moulton A., Madnick M., and Siegel M., (2003), "Context Interchange Mediation for Semantic Interoperability and Dynamic Integration of Autonomous Information Sources in the Fixed Income Securities Industry", No 4404-02 Working papers from MIT Sloan School of Management

DSouza, (2001)," Model-Driven Architecture and Integration Opportunities and Challenges", Kinetium 2001

Sergey Melnik, (1999), "Generic Interoperability Framework", Distributed and Parallel Databases Journal 1999

D. Oberle, S. Staab and A. Eberhart, (2006), "Semantic Management of Distributed Web Applications", IEEE Distributed Systems Online, vol. 7, no. 4, 2006, art. no. 0605-o5001

Qin, J. and Prado, J.C., (2006), "The semantic and syntactic model of metadata", Série Comunicação da Informacao Digital 2006 Vol. 4 (pp. 143-156)

Qin, J. and N. Hernandez., (2004), "Ontological representation of learning objects: building interoperable vocabulary and structures", ACM Press New York, May 17-22, 2004, 348-349

Davis L., Gamble R., Payton J., Jónsdóttir G. and Underwood J., (2001), "A Notation for Problematic Architecture Interactions", 9th Int'l Symposium of Software Engineering (ESEC/FSE-9) pp. 132-141

F. Varvey, (1999), "Semantic Interoperability and Citizen/ Government Interaction", University of Minnesota (working paper)

Cambrin D., Demeule L., Dumedah G., Hoare D. And Simmonds K., (2004), "Sub-Surface Data Management", Geography 455 Advanced Issues in GIS March 2004

Aberer K., Cudré-Mauroux P. and Hauswirth M., (2002), "A Framework for Semantic Gossiping", NCCR-MICS the Swiss National Science Foundation 5005-67332

Mutton P. and Golbeck J., (2003), "Visualization of semantic metadata and ontologies", Seventh Int'l Conference on Information Visualization (IV03), pages 300-305. IEEE, July 2003

Panfilis S. and Berrel A., (2001), "Open issues and concerns on Component Based Software Engineering", CBSEnet publication IST-2001-35485

Pierre M. and LaPlant W., (1998), "Issues in Crosswalking Content Metadata Standards", National Information Standard Organization 1998

M. Staab, N. Stojanovic, R. Studer, and Y. Sure, (2001), "Semantic Portal - The SEAL approach. to appear in: Creating the Semantic Web.", MIT Press

O. Valkeapää and E. Hyvönen, (2006), "Semantic Annotation with Browser-based Annotation Tool", 1st Asian Semantic Web Conference

R. Barcia and J. Brent, (2005), "Building SOA solutions with the Service Component Architecture", IBM Research

M. Schmalz, (2003), "EITH – A Unifying Representation for Database Schema and Application Code in Enterprise Knowledge Extraction" , 22nd International Conference on Conceptual Modeling

Copyright © 2007 Inderscience Enterprises Ltd.

O. Zimmermann, P. Krogdahl, C. Gee, (2004), "Elements of Service-Oriented Analysis and Design", IBM Academy

Sciore E., Siegel M. and Rosenthal A. , (1994), "Using semantic values to facilitate interoperability among hetero geneous information systems", ACM Transactions on Database Systems v.19 I.2

M. Staab, N. Stojanovic, R. Studer, and Y. Sure, (2001). "Semantic Portal - The SEAL approach. to appear in: Creating the Semantic Web.", MIT Press

O. Valkeapää and E. Hyvönen, (2006), "Semantic Annotation with Browser-based Annotation Tool", 1st Asian Semantic Web Conference

O. Zimmermann, P. Krogdahl, C. Gee, (2004), "Elements of Service-Oriented Analysis and Design", IBM Academy

Copyright © 2007 Inderscience Enterprises Ltd.

T3: Micro Process Design in Component Based Architecture

Publication: Haamaha July 2007

International Conference on Human Aspect in Advanced Manufacturing and Hybrid Automation

9th-12th July, 2007, Poznan, Poland (http://www.ergonet.net/haamaha2007)

Publication Contact: Prof. Stefan Trzcieli

Chairman, Haamaha 2007

Vice Dean of Computing and Mgt

Poznan University of Technology,

Stefan.Trzcielinski@put.poznan.pl



From: "HAAMAHA 2007" <haamaha@put.poznan.pl>

To: rwu_academic@yahoo.com

Subject: Re: HA5K066

Date: Wed, 7 Mar 2007 19:16:34 +0100

Confirmation and invitation for participation at International Conference on Human Aspect in Advanced Manufacturing and Hybrid Automation " Managing Enterprise of the Future " July 9 th – 12 th , 2007. Poznan University of Technology ,

Dear Mr RAYMOND WU ,

RE: HA5K066 Micro Process Design in Component based Architecture Integration

Congratulations! I am pleased to inform you that your paper(s) has been accepted for presentation at HAAMAHA ' 07. If you have co-authors whose e-mail addresses were not provided earlier, Poznan , Poland.

Sincerely Yours,

Stefan Trzcielinski

Stefan Trzcieliński, Ph.D., D.Sc.

Associate Professor Vice Dean of Faculty of Computing and Management Poznan University of Technology, Institute of Management Engineering

ttware system
emission from

Otganizations in Changing Environment. Current Problems,Concepts and Methods of Management

Proceedings of the Eleventh International Conference on Human Aspects of Advanced Manufacturing, Agility and Hybrid Automation
4th International Conference ERGON-AXIA

HAAMAHA 2007

Managing Enterprise of the Future

July 9-12, 2007 Poznań, Poland

Edited by:

Wiesław M. Grudzewski
Irena Hejduk
Institute of Organization and Management in Industry

Stefan Trzcieliński Poznań
University of Technology

o persons or
ise or
operarch.

2007

IEA Press International Ergonomics
Association 610 Walnut Street 575
WARF Madison, WI 53726 USA

Newform Enterprise

1 A framework [illegible] the e-work service for future workers Jung Jong Myung, Asian Jung Hyun, Mat Kyoung Seon, Lee Dong Myun, Kim Sang Ha
2 The impact of [illegible] on firm performance
[illegible]
Sudarsan Chandra, Gupta Ritinder, Sahoodin al Sany, Hassan Yusf
3 Influence of [illegible] on [illegible]
Radzicki Tomasz, Zasadzki Grzegorz
4 Micro process design in component based architecture
Wu Raymond
5 Enterprise Architecture evaluation [illegible]
Niemi Esko, Ylimäki Tanja
6 The decision and analysis of component based integration
Wu Raymond
7 The role of architecture evaluations in ICT companies Hindle Noora, Ylimäki Tanja, Niemi Esko
8 Enterprise architecture process of a telecommunication company a case study on virtualization
Arslanlar Ali, Hamamcıoğlu Nesa
9 Design and information technologies
tools for innovation's processes in Mexico
Meza Aguilar Jorge
10 Enterprise component based interoperability in e-government
Wu Raymond
11 Intelligent agents to support e-business applications
Olek Jr. Tomasz, Kozek Marek
12 Complex biometric algorithms for people identification
Dębowski Adam, Marciniak Tomasz, Dąbek Szymon
13 The impact of web services on companies
Kassina Tarmo-Kaie, Tiidorg Vesse-Aleksei, Beshpalko Sergei, Chaves Sandra, Koppel Rosalia
14 The internet in polish small and medium-sized enterprises Arcelus Teresa Annotating multimedia documents with multimedia document in a decision support system Charles Robert Hicks, measure work time losses

Micro Process Design in Component based Architecture

Raymond Wu
IBM Software Group
+49 7031 16 1755 / raymond@de.ibm.com

Decision, analysis and design are key steps in building the skeleton of enterprise integration. In this research paper we deal with the design stage by introducing the micro process and interconnectivity concept as the foundation of the service component design. This work is based on the common component concept we implemented in previous research and on the component-based design foundation we build in the current research.

Keywords

Integration, Component, Enterprise, Interoperability, Semantic, Mediation

Introduction

Our research in enterprise integration has been divided into four areas: decision, analysis, design and virtualization. In our previous research, political model decision-making and rational model analysis were applied to the key areas in which the components of business, service and physical elements are selected and validated in provisioning a particular business context goal or KPI within the project scope of the enterprise integration dimension. In the current research work, we shift our focus to the design stage to examine how the components are designed as atomic units, how they work together for interconnectivity and how they achieve the enterprise goal in vertical interoperability. As the architecture design and configuration is defined in the analysis stage, our focus of design is the micro processes of component manufacturing, interconnectivity between components and micro process design of service provisioning.

The micro process of enterprise integration differentiates itself from the macro process in that the macro process is working on the configuration of components at an architectural level while the micro process deals with components both internally and in their interference in component fabrication, with how they are integrated in micro mechanisms and with how they work together in provisioning the coherence of a business goal. This includes the integration of common components and service aggregation in micro processes.

Both micro and macro processes are important to enterprise integration. Our integration strategy and framework presented in a separate research paper indicated that the transformation from business context to service component cannot be covered solely by the deconstruction process; the business semantic injection in micro process is crucial due to each component being responsible for a particular business rule. In a component-based implementation there is no duplication or mix-up allowed in component design; if the dependency between particular components is high and they always follow the same

sequence in their referential integrity, we call this a good candidate for service aggregation. Those components with a high frequency of utilization are termed common components. Both aggregated and common components need to be handled carefully; the steps for achieving this level are shown in Figure 1.

c1. At an enterprise level exceptions in handling a particular requirement are normally not allowed in order to maintain high integrity within the enterprise; therefore, consensus and enterprise standards are usually required at the outset prior to the design.

c2. As aggregated services and common components are critical in forming the enterprise skeleton, the design and aggregation process must be cautious from an interconnectivity perspective. This implies that high integrity, optimization, grain size, dimension in service provisioning, visibility, traceability, performance and other factors are areas which must be brought into consideration.

c3. Once all of these skeleton elements are well specified and designed, the "main route" needs to be outlined at a state-of-the-art level. What we term the main route is based on the top of the enterprise skeleton we build; the value-added service may mediate this aggregation and form a long-running process as a skeleton for a business line. One example is in the credit card business in which aggregated services have been built between the card holder and card issuer; in this case the same aggregation is also being formed between the merchant and the bank. The mediation service of an inter-banking fraud detector component has been designed by reading fraud profiles in real-time transaction environments to mediate the individual services into a business main route; this highly enhances the service provisioning in a particular dimension.

c4. Aggregated services and common components should not be tightly coupled; instead, their dependency and interconnectivity should be based at a parameter and metadata level and their design and interfacing specifications should be fully visible for future development. The reason for this is that the more significant role these skeleton elements play, the higher the complexity of interconnectivity, business criticality and access frequency, which can result in significant impact if a defect is detected too late. In many instances, a defect is a kind of binding between components instead of a coordinate from aggregation.

c5. A service aggregation needs to be implemented in an empirical rule-base in the same way that a phrase needs words, grammar, semantics and a certain level of common recognition. It should not be allowed to be handled for individual needs differently or randomly; this is the reason it should be designed and maintained at the enterprise level as a single view of enterprise solution.

Desmond DSouza defines vertical integration in his "Model-Driven Architecture and Integration Opportunities and Challenges," stating that vertical integration comes from different levels of abstraction of the same system, from physical data models, logical data models, networks, applications specifications, component assemblies, business process models, and business goals and strategies [1].

A component-based model builds the foundation for both business and service components, storing the intelligence of mapping and interconnectivity information in a repository in metadata form and strengthening the foundation of vertical enterprise integration.

Before we extend our findings into componentization, we will define the term component as follows:

According to P. Kruchten, a component is a non-trivial, nearly independent and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture. A

component conforms to and provides the physical realization of a set of interfaces. First, a component is non-trivial; it is functionally and conceptually larger than a single class or a single line of code. Typically, a component encompasses the structure and behavior of a collaboration of classes. Second, a component is nearly independent of other components. It rarely stands alone. A given component collaborates with other components and in so doing assumes a specific architectural context. This architectural context is driven in large part by the implementation we chose. Third, a component is a replaceable part of a system. A component is substitutable for any other component that realizes the same interfaces. This aspect helps during development in which parts of a system can be stubbed, sketched and then replaced by mature, robust implementations. It also supports the evolution of a system even after it has been deployed by making it possible to upgrade and evolve parts of the system independently. Fourth, a component fulfills a clear function. It is logically and physically cohesive and thus denotes a meaningful structural and/or behavioral chunk of a larger system. It is by no means only an arbitrary grouping. Fifth, a component exists in the context of a well-defined architecture. A component represents a fundamental building block upon which systems can be designed and composed. This definition is recursive: a system at one level of abstraction may simply be a component at a higher level of abstraction, yet components never stand alone. Every component presupposes an architectural and technology context in which it is intended to be used. Finally, a component conforms to a set of interfaces. A component that conforms to a given interface means that it satisfies the contract specified by that interface and may be substituted in any context to which that interface applies. [2]



The micro process starts with the mechanism for how the component is designed; this involves taking the business semantic of the macro level with coherence of enterprise interoperability between each layer. The service component possesses the capability of semantic handling, interfacing and metadata integration as shown in Figure 2.

Micro process in component transformation within the service layer

Before the design of the service component, we take direction from the business component by using a top-down approach to derive the component design while at the same time streamlining the alignment between service components and physical components by using both top-down and bottom-up approaches. Most of the bottom-up approaches are a componentization process working from existing information components, business objects, codes, system metadata or database access protocols. Once we construct the alignment between business, service and information and have the service component design and interconnectivity completed, we actively work on service layer transformation. The reason this step is crucial is that the goal of vertical enterprise integration is to create interoperability between layers from both micro and macro perspectives, with both business and information layers working as a foundation to support the formation of the service layer which provides the actual services. Once the service component has completed its design with alignment to business and information, the focus can be shifted to micro interference within the macro view.

In our previous research on metadata interoperability we revealed, in relation to both political and rational models, how the components are derived through the validation process and how the strategy of component implementation achieves layer interoperability. In this paper we extend our research in component design and technical implementation for both internal and interface component factors. [3]

Micro interoperability in enterprise integration and service virtualization

Common service and common information is a top-down approach which follows the nature of business implementation direction. From the technical perspective the idea was to eliminate the integration effort by using the "virtual organization" concept. From the organization perspective this enhances the flexibility of service provisioning and the agility between organization and service, thus increasing the interoperability across organizations. The realization of this top-down vertical integration approach requires robust middleware to support the inter-communication between component-and-component and component-and-repository. In other words, the concept needs physical support from an integration service bus such as the middleware in run-time services.

The component element is the physical element that mediates service and information. The service component consists of interface protocol, business rules, semantic intelligence and referential information. It is important to note that (a) components are encapsulated for reuse; (b) components may be wired together as composite services; and (c) business objects are the data flowing on wires between components.

The design of service components follows our previous strategy by using business semantic injection to take the context into the semantic and micro processes; by this means each component has its contract level responsibility in provisioning a subset of service. This interface helps the coordination between components and metadata to build the foundation of interoperability within the service layer and between layers. In the following paragraph we take the results from component design into the service level while applying automata theory to build up the rule-based service aggregation at the enterprise level. As we are still at a pioneer stage in applying automata theory in service provisioning and aggregation, we understand this cannot be accomplished within a short period. Further research efforts will be undertaken following the current work. [4]

SCA (service component architecture) of the service component

SCA deals with component design and interconnectivity in micro mechanism and consists of three main areas. As shown in Figure 3, the service component is produced in the following three criteria: [5]

1. Interface

The service interface specifies SP (service provider) implementation and is invoked by the SC (service coordinator). The SC has the capability to invoke SPs through the service bus without needing to know what SPs are and how the interface was configured.

Luca et al. demonstrated their surveying results from the theory behind formalizing rich interfaces for software and hardware components. Rich interfaces specify the protocol aspects of component interaction. Their formalization, called interface automata, permits the compiler to check the compatibility of component interaction protocols. Interface automata support incremental design and independent implementability. Incremental design means that the compatibility checking of interfaces can proceed for partial system descriptions without knowledge of the interfaces of all components. Independent implementability means that compatible interfaces can be refined separately while still maintaining compatibility. We used both of these methodologies in component interface design. [6]

2. Implementation

The service provider implementations in an SCA are called service components that include:

- **Java objects**

A Java object is an implementation of a component in Java which has either a WSDL (web service definition language) or java interface type. A Java object normally works in a mediation module.

- **Business processes**

A business process is a combination service made up of service components with a specific order to achieve a business goal; a business process should not be equal to a use case, rather it can be a result of decomposition and composed of use case.

- **Variables**

Variables store the data that are used by a business process. The messages represent the writing and the paper represents the variable.

- **Microflow or short running process**

A microflow is a process that is contained within a single transaction. This is ideal for situations where the user is expecting an immediate response.

- **Long run processes**

A long running process executes over an extended period of time and is much more flexible and resilient than a microflow. Interruptible business processes and asynchronous business processes are examples of long running processes.

- **Human tasks**

A human task is a unit of work done by a human which involves the interaction with other services; it thus becomes a task within a larger business goal.

- **Business state machines**

A state machine is an event-driven business transaction in which external operations trigger changes that guide the transaction from one discrete mode to another. Each mode is an individual state, and this mode determines what activities and operations can occur.

- **States**

A state is one of several discrete individual stages that are organized in sequence to compose a business transaction.

- **Composite state**

A composite state is an aggregate of one or more states. It is used to decompose a complex state machine map into an easily comprehended hierarchy of state machines.

- **Business rules**

A business rule is anything that captures and implements business policies and practices. A rule can enforce business policy, make a decision, or infer new data from existing data.

3. Metadata

The parameters transmitted in and out from a service are modeled as service data objects implemented by Java class. SDO (service data object) complements SCA by providing a common way to access many different kinds of data. The specification reduces the skill levels and the time required to access and manipulate business data. Today, a great many APIs are used to manipulate data. These APIs tend to tightly couple the source and target of the data; this makes their usage error-prone and subject to breaking as business requirements evolve. However, SDO makes them easier to use and thus realize the value of these APIs without having to code directly to them.

Semantic event aggregator by using automata theory

1. At the very beginning of a use case, the first event triggers service Φ_1

$\varepsilon^1 : r_0 \times \Phi_1 \rightarrow r_1$

2. When the service event Φi ($i <> 1$) is read and triggers the service component, we express the state transformation as follows:

$\varepsilon^i : r_{i-1} \times \Phi_i \rightarrow r_i$, we further derive the transformation as follows:

$\varepsilon^{i+1} = (r_i) \times \Phi_{i+1} \rightarrow r_{i+1} = (r_{i-1} \times \Phi_i) \times \Phi_{i+1} \rightarrow r_{i+1}$

which is in equilibrium to $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$, and derives

$\varepsilon^{nm} : r_n \times (\Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m) \rightarrow r_{m\ (n<m)}$, we call this the "semantic event aggregator."

3. As the final step of a use case, the final event trigger service Φ_n

$\varepsilon^n : r_{n-1} \times \Phi_n \rightarrow r_n$. Φ_n is a terminator, an example of which is the settlement of a daily transaction.

In any component context a formal interface contract for the invocation syntax is important. The semantics issue (the meaning of parameters and so forth) has to be solved as well (domain modeling). This is crucial to any business-to-business (B2B) and dynamic invocation scenario. Such scenarios are cornerstones of the service component vision of being flexible and agile in response to the new business needs in a world of mergers and acquisitions, business transformation, globalization and other relevant factors. [7]

A tool called JFLAP guides the user through several proofs of theorems that focus on transformations of languages from one form to another in which the forms are automata and grammars. JFLAP aids in automata visibility; users are given a visual representation they can interact with in addition to the formal textual representation. This alternative hands-on representation provides a means for an interactive automata theory. [8]

The string $\Phi (r_n , r_m) = \Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m$ is a path from r_n to r_m, we call the "service route" and $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$ a "composite service" between r_{i-1} and r_{i+1}, we use the symbol $\Phi_{i, i+1}$ to denote a composite service.

When frequency $F(\Phi_{i, i+1}) = F(\Phi_i \times \Phi_{i+1})$ is higher than threshold σ, we aggregate the concatenated alphabet (part of the string) $\Phi_i \times \Phi_{i+1}$ into $\Phi_{i , i+1}$, this implies that the original state r_i between r_{i-1} and r_{i+1} will no longer exist for this composite service.

Main route

When the business architect checks the result from semantic event aggregator - ε^m which is more an upper-layer use case scenario, service route - $\Phi_{i, i+1}$ which is in the service layer and its frequency - $F(\Phi_{i, i+1})$, if $F(\Phi_{i, i+1})$ is much higher than threshold σ, then from the business perspective the event aggregator ε^m either consists of many composite services or is business critical. This means the aggregation service is the key within business and is crucial in daily operation from service perspectives and that fine tuning is feasible from a multi-fine grained boundary to a coarse grained service. We call this the "main route" of the enterprise skeleton; in other words this is the highway between A and B. Virtual coarse grained service eliminates the boundary and leverages many fine-grained components which can be easily configured by adding or modifying the component dynamically to achieve the broader coverage of main route services. For this reason main route is one of crucial services in enterprise integration. One example is in the Telecommunication industry that incorporates an open service account, an open billing account and provisioning options for a new customer. This constitutes a main route for "new customer service" coarse grained service, though it is aligned with fine-

grained components to support customer preference, service criteria and billing options which are branches within this main route.

Alternative service shortcut

In service route $\varepsilon^{nm}: r_n \times (\Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m) \rightarrow r_{m,\ (n<m)}$ if these exist in a composite service Φ_i that r_i is omitted ($n < i < m$). All the above terminology is shown in Figure 4 regarding the formation of composite, alternatives and main route. All of these critical options are enabled by the automata model as expressed in Figure 5.

Common interface

Common interface is also a means to achieve common components. S. Melnik proposes a universal interface which avoids common models and languages; this is the Generic Interoperability Framework which was developed to facilitate the integration of heterogeneous information systems. Within this framework individual communication protocols, data manipulation languages and data are represented in a generic manner while preserving their ontological variety; this process indicates how a mediation infrastructure can be built using canonical wrappers, operational requirements of integration and layered architecture for component design [9].

Enterprise integration

The importance of determining the common component and service aggregation is in building up an enterprise skeleton of vertical interoperability between business, service and information. This task is crucial, as in the business domain analysis stage, the impact area and scope to business components are minimized in decision making. Additionally, the business component is deconstructed into service components and physical elements for top-down alignment while the common component is not simply a symbolic service but instead requires a bottom-up approach to work on common areas; this serves to fine tune the common components and interfaces from the perspective of enterprise integrity and performance. The service aggregation is also important as the top-down deconstruction followed by bottom-up composition forms the skeleton of vertical interoperability and potential component-based implementation of the service process.

The iteration through each use case will be tracked for its path and service statistics so that once we parse all use cases we should be able to see the frequency of use of each service as well as the path and dependency between services. Where there is a high frequency of services we use the term "common services" and focus on the potential for reuse and minimization of the integration interface. In the case of service paths in which several services have a common sequence and appear in use cases a certain number of times past the threshold, the services can be aggregated to a composite service or potential process. This needs to be designed at an enterprise level so that each organization has the same concept of and specification for this common service. Service aggregation means a composite component consisting of at least two components that form a unique sequence of execution order with a common interface within their boundaries; this composite component will be treated as a coarse grained service in provisioning the upward aggregation of services.

Service mediation patterns

Mediation patterns manipulate messages in-flight on the bus by requests or events. According to IBM Service mediation, there are several basic patterns for mediations; the following more complex patterns can be built by combining the simple patterns:

- **Protocol switch:** Enables service requesters to dispatch their messages using a variety of interaction protocols or APIs such as SOAP/HTTP, JMS, and the MQ Integrator (MQI).
- **Transform:** Translates the message payload (content) from the requester's schema to the provider's schema. May include enveloping, de-enveloping, or encryption.
- **Enrich:** Augments the message payload by adding information from external data sources such as customization parameters defined by the mediation or from database queries.
- **Route:** Changes the route of a message, selecting among service providers that support the requester's intent. Selection criteria can include message content and context as well as a target's capabilities.
- **Distribute:** Distributes the message to a set of interested parties and is usually driven by subscriber interest profiles.
- **Monitor:** Observes messages as they pass through the mediation unchanged. Can be used to monitor service levels, assist in problem determination or meter usage for subsequent billing to users.
- **Correlate:** Derives complex events from message or event streams. Includes rules for pattern identification and rules that react to pattern discovery; for example, by generating a complex event derived from the content of the triggering event stream.

Mediations can be explicitly configured within a solution. As well as being attributes of the services, policies can be set by the solution administrator for a single interaction or set of interactions. For example, it can be set to log all messages to a particular external provider or with a transaction value greater than US$1 million. [10]

D. Roman et al. in their "Web Service Modeling Ontology (WSMO)" state that semantic services need to meet the following requirements: Web Compliance, Ontology-Based, Strict Decoupling, Centrality of Mediation, Ontological Role Separation, and Execution Semantics. [11] These requirements were addressed in our research and the strategy is identical to that which we proposed.

Service virtualization

Once the vertical skeleton and service extension are built, the interoperability reflected from the business layer has the capability of integrating cross-organization and cross-geographical constraints, as fully transparent services become possible if each system or organization follows an industry standard by using metadata strategy and an industry framework. An example of this would be a case in which several key industry vendors have built up an industry framework for service providers in that particular industry to follow, starting from business domain analysis and working through to component deployment. Each service provider would thus benefit from the industry common services and the common interface, particularly those involving merger and acquire cases (B2B, B2C). Any business activity becomes easier when the various parties involved have a common understanding and a consensus on business terms, service definitions, interfaces and component specification, particularly if these become transparent and common practices. The important concept here is that once the metadata of business, service and information constructs an industry standard, the organization's vertical skeleton has the opportunity to integrate across multiple organizations for an easier global practice; this is a long-term industry objective which we call service virtualization in a fully transparent environment.

The interoperability skeleton shown in Figure 6 is an example of how the micro interoperability in the service layer, aligned with the component layer in micro mechanism, provisions the macro service in business. The foundation of this skeleton is achieving interconnectivity between service components by using a common component

and an aggregated component. We further develop the service virtualization on top of this interoperability skeleton, which we call enterprise vertical integration, in extending the integration across industry and geographical areas. This is the virtualization service in a horizontal direction; once the industries of different regions build up their semantic interoperability in three metadata repositories (business, service and information), they can synchronize their business, operation and strategy even if they have their local implementation extended from the interoperability skeleton.

Conclusion

Enterprise interoperability is built on the foundation of a validation process of decision, design and provisioning stages. In this paper we focused our research on micro process design; this encompasses component-based design and interconnectivity, transformation of service aggregation through rule-based design, service provisioning in a particular dimension and service virtualization.

In component design we introduced the SCA (service component architecture) to further clarify the micro mechanisms within components as well as component interconnectivity. We classified component design into interface, implementation and metadata areas to guide the design from the internal to external.

In transformation of service aggregation we explored the possibility of applying automata theory in service aggregation. We found that the concept of automata theory can be applied to service aggregation in building up rule-based aggregated services if we handle the service execution by using the same concept of symbolic reading. Handling the parser in use case parsing by using the same machine reading concept, the state concept can also be applied to the state of service layer with metadata alignment to the business and information layers.

In enterprise service provisioning and virtualization, we constructed the skeleton of enterprise service with leverage to each layer and the interoperability of the layers. Service virtualization is a process of leveraging enterprise integration strategy within a specific geographical region or across regions. This can be an industry standard scenario such as a credit card transaction in which the service, geography and systems are fully transparent to both cardholders and merchants. A full virtualization environment needs to be built on top of the foundation of component design and service transformation process in order to apply the vertical enterprise integration approach to this final stage which is the goal of our research work. The standardization of metadata technology and common services and components are all requirements in this stage.

Certain areas in this research, such as applying automata theory in forming aggregated services, are still in the pioneer stage; we will extend our research in the near future to provide more guidelines for academic workers who are interested in this area.

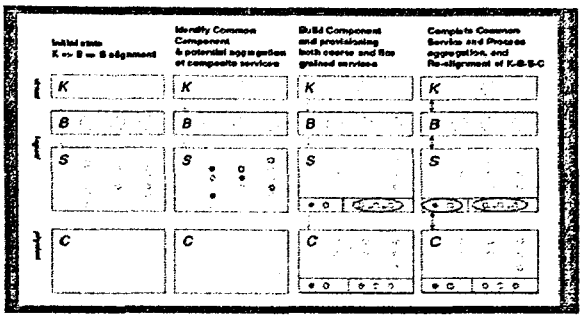


Figure 1. K-B-S-C interoperability and alignment

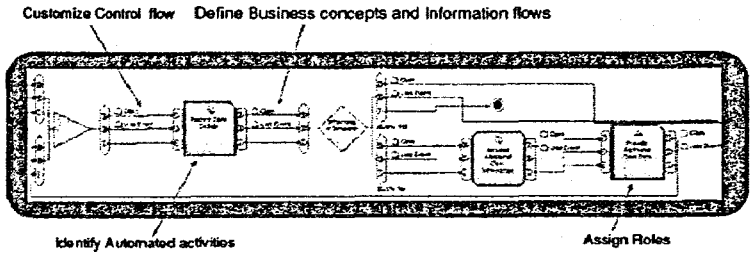


Figure 2. SCA (Service component architecture) of service component

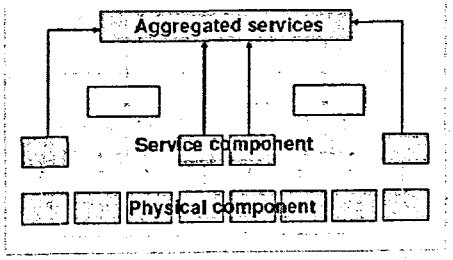


Figure 3. Aggregated service in common services

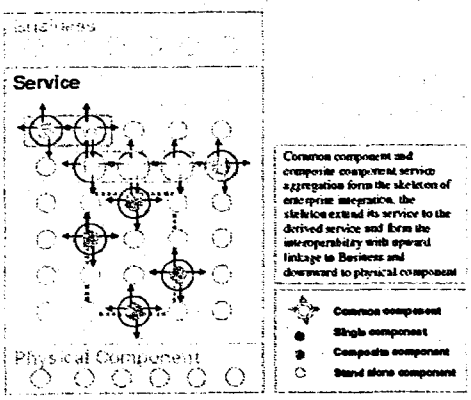


Figure 4. Skeleton of enterprise integration interoperability

We thank reader in reading our publication, specials thanks to The 11th International Conference on Human Aspects of Advanced Manufacturing Agility and Hybrid Automation in publishing our research work.

1. DSouza,(2001)," Model-Driven Architecture and Integration Opportunities and Challenges", Kinetium 2001

2. Philippe Kruchten, (1998), "Modeling Component Systems with the Unified Modeling Language", 1998 International Workshop on Component-Based Software Engineering

3. Jie Lu, Guangquan Zhang & Fengjie Wu, (2005), "Web-based Multi-Criteria Group Decision Support System with Linguistic Term Processing Function", IEEE Intelligent Informatics Bulletin June 2005 Vol.5 No.1

4. R. Barcia and J. Brent, (2005), "Building SOA solutions with the Service Component Architecture", IBM Research

5. Raymond Wu., (2006), "Enterprise Integration in semantic metadata", IEEE (IT professional) ITPro-0055-0806

6. L. de Alfaro and T. A. Henzinger, "Interface theories for component-based design," in *Proceedings of theFirst International Workshop on Embedded Software* (EMSOFT), Lecture Notes in Computer Science 2211,Springer-Verlag, 2001, pp. 148-165.

7. O. Zimmermann, P. Krogdahl, C. Gee, (2004), "Elements of Service-Oriented Analysis and Design", IBM Academy

8. E. Gramond and S. H. Rodger, (1999), "Using JFLAP to Interact with Theorems in Automata Theory", Thirtieth SIGCSE Technical Symposium on Computer Science Education, ACM Press 1999 page 336-340.

9. Sergey Melnik, (1999), "Generic Interoperability Framework", Distributed and Parallel Databases Journal 1999

10. B. Hutchison and M. Schmidt and D. Wolfson, (2005), "SOA programming model for implementing Web services", IBM Research 2005

11. D Roman et al, (2005), "Web Service Modeling Ontology (WSMO)", European Commission under the projects DIP 2005

T4

T4: Implementation Priority and Dependency in Component Based Design

Publication: EISWT/ IEEE CS Proceeding (http://www.promoteresearch.org/2007/eiswt/index.html)
International Conference on Enterprise Information Systems and Web Technologies (EISWT-07)
July 9-12, 2007 Orlando, FL, 32819-8284, USA.

Publication Contact: Prof. John E. Edward
Editor in chief, EISWT
Montana State University
jeedward@gmail.com



"John Edward" <jeedward@gmail.com> Add to Address Book Add M

From: Yahoo! DomainKeys has confirmed that this message was sent by gmail.com more

To: nwu_academic@yahoo.com

Subject: EISWT-07

Date: Fri, 19 Jan 2007 18:56:53 -0500

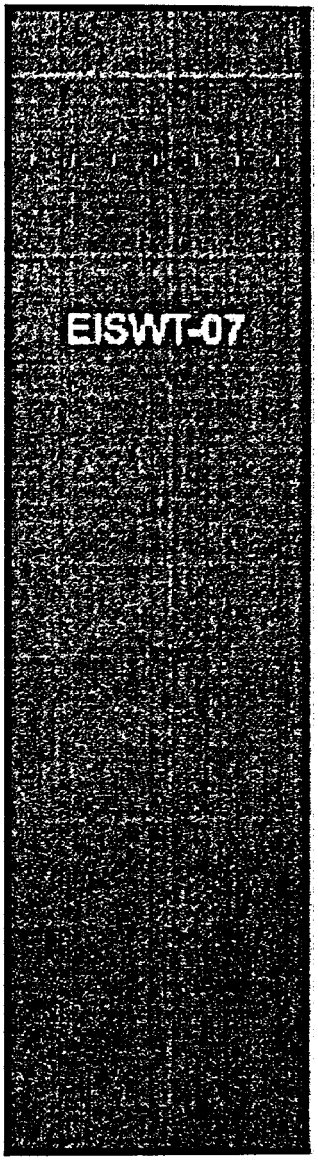
Dear Colleague(s)

On behalf of the committee of the 2007 International Conference on Ent Information Systems and Web Technologies (EISWT-07), I am pleased t you that your paper: Implementation priority and dependency in compon based design
Raymond, Cheng-Yi Wu
IBM Germany
Faculty of Information Technology
University of Technology, Sydney
P.O. Box 123, Broadway, NSW 2007, Australia

ID: eiswt106
has been accepted for presentation at the EISWT-07 conference to b Orlando, FL, USA during July 9-12 2007. Congratulations! The paper be included in the proceedings. The reviewers' comments are attache email. The conference organizers should receive the revised and formatted final paper, copyright form and the registration fee on or bef 2nd 2007, in order to include your paper in the conference proceedings.

Sincerely

John Edward
(on behalf of EISWT-07 committee)



PROCEEDINGS OF THE 2007 INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS AND WEB TECHNOLOGIES

Editors:
Bobby Granville, N.S.Kutti,
Michele Missikoff and
Ngoc Thanh Nguyen

Associate Editors:
Scott Graham, Naphtali D. Rishe
and Boosern Wongsaroj

Orlando, Florida, USA
July 9-12 2007

©ISRST 2007

ISBN: 978-0-9727412-4-6

Contents

SESSION: Databases and XML
SESSION CO-CHAIRS: Scott Graham, Naphtali D. Rishe and Booseren Wongsaroj

Implementation Priority and Dependency in Component Based Design 42
Raymond Wu

2007 International Conference on Enterprise Information Systems and Web Technologies (EISWT-07)

Implementation priority and dependency in component-based design

Raymond Wu
IBM Software Group
raymond@de.ibm.com

Abstract

In component-based design, we analyze the components of business, service and physical elements based on their priority and dependency. Common components are those which have priority and are frequently used, and dependency has to do with the consolidation of aggregation from fine-grained components in provisioning coarse-grained services. We applied parsing techniques in use case simulation and dependency analysis to produce the common services and aggregation services. Common service and aggregation is determined by a rational model instead of by a political decision model. We plot the component evaluation against business, information, component and aggregation maps to access their tendency and intensity, and we are then ready to construct the skeleton and assign priority in component implementation.

Introduction

There are six steps in implementation and realization in component-based design:

1), 2) and 3) Validation of business, service and physical components

4) and 5) Common component and service aggregation parsing analysis

6) Determining priority and skeleton within enterprise integration

This research paper covers Steps 4 and 5 but will focus mainly on Step 6. In conventional development, weak visibility between business, service and information makes linkage and enterprise interoperability impossible. In order to shed light on common components and their interoperability, we introduced our approaches in creating visibility and identifying each component's micro process design. Our strategy of enhancement is ordered as follows:

i) Componentization and creating linkage between each layer

ii) Dimensional implementation driven by business scenario

iii) Creating common services foundation using a bottom-up approach

iv) Top-down parsing for common services in each particular dimension

v) Identifying common components and aggregation by parsing techniques

vi) Further development of interoperability between top-down and bottom-up

vii) Customizing and provisioning services in the virtualization environment

Common service is a conceptual term with alignment to both business components and physical objects; these can be either functional or non-functional elements. Functional elements are produced by a top-down approach while non-functional elements are produced by a bottom-up approach. The service aggregation parsing process identifies these two services as candidates for common service; they are validated once the frequency passes the appropriate threshold and their interconnectivity design is seen as feasible. Our approach in producing priority and dependency is based on the deconstruction from both business components which we identified as 'active map' business driven and infrastructure level non-functional elements. As each service component and physical element is an atomic component which we call finest grained, we identify those frequently used fine-grained components as common components. We differentiate 'common' and 'non-common' by validating their priority and design details; a common component is a candidate for enterprise skeleton element and aggregation services as we need to build the service foundation before extend the implementation to non-common peripherals. These are shown in Figure 1.

Literature review

Metadata interoperability attempts to homogenize disparities through standards. The interest of many researchers regarding semantic interoperability is how people and social groups with different perspectives identify and possibly resolve their semantic differences (Harvey, 1999) [1]. As far as

we know, the construction of information communities in heterogeneous institutional and disciplinary environments calls for frameworks to conceptualize and articulate these semantic differences. A reason for this is that a geographical heterogeneous state in semantics forces peers to follow a power law distribution which emerges frequently in networked interactions [2].

importance of "events" to unambiguously describe metadata for objects with a complex history. To test and evaluate the interoperability capabilities of this model, some real multimedia examples have been applied to analyze the results of mapping from this data model to a variety of different metadata domains.

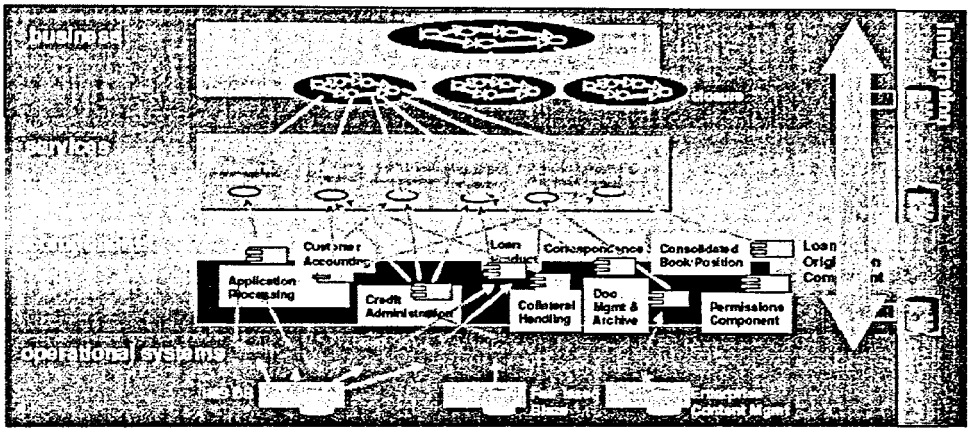


Figure 1. Top-down and bottom-up approaches in producing service components

The other concern is that implicit information embedded in semantic web graphs, such as topography, clusters and disconnected sub graphs, is difficult to extract from text files [3].

According to L. Yu et al., an obstacle to software reuse is the large number of major modifications that frequently have to be made as a consequence of dependencies within the reused software components. Component-based implementation is the foundation for coping with this bottleneck [4]. Therefore, the determination of common and composite services is a use case driven parsing process; the purpose of the parsing iteration process is to discover the common areas and dependency between services.

J. Hunter argued that metadata interoperability is a fundamental requirement for access to information within networked knowledge organization systems. For this purpose, a common underlying data model has been developed to enable the scalable mapping of metadata descriptions across domains and media types. This model provides a set of basic building blocks for metadata modeling and recognizes the

To overcome the limitations in semantic mapping, the metadata "thesaurus," capable of providing additional semantic knowledge, has been developed to enable flexible and dynamic mapping among metadata standards [5].

Priority and dependency in the rational model

In each K implementation, the interoperability is built by using semantic business deconstruction to derive service elements and component elements; at this point elements can be aggregated into composite components or identified as a common service for reuse and shared service purposes. There are three layers of assessment in determining the tendency of each service component.

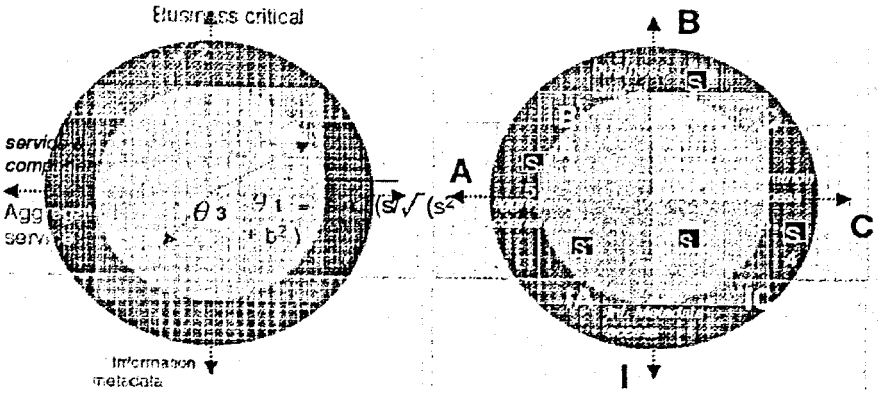


Figure 2. Intensity and tendency analysis of B-C-I-A

- Business layer - Group decision to determine which business component to use
- Service layer - Group decision with some analysis work from architecture
- Component layer - Group decision with some analysis work; a pre-built information foundation in meta form is required to make sure the information metadata is well integrated in this layer
- Service transformation – Identify services in relation to common and aggregated components

Let $\Phi^*(SC^k_i)$ denote $(\Phi^*(S^k_i) + \Phi^*(C^k_i))$ $(1 < i < n)$, and represent the vertical intensity of service component i (under strategy k) in both upward business and downward information dimensions; $\Phi^*(SC^k_i)$ is actually a matrix produced by the political model. Meanwhile, we introduce $\Psi(S^k_{ij})$ to denote the matrix of counts from our use case parser

$$\text{Let } \Psi(S^k_{ij}) = \begin{matrix} \\ (1<i<n,\ 1<j<n) \end{matrix} \begin{bmatrix} \gamma(S^k_{11}), \gamma(S^k_{12}), \ldots \gamma(S^k_{n1}) \\ \gamma(S^k_{12}), \gamma(S^k_{22}), \ldots \gamma(S^k_{n2}) \\ \cdots \\ \gamma(S^k_{1n}), \gamma(S^k_{2n}), \ldots \gamma(S^k_{nn}) \end{bmatrix}$$

Once we finish the design stage and before we implement the component into our scope of project, the following measurement is used in evaluating the intensity of the component, employing its tendency as a reference.

The method used is to partition the domain space into three zones: business, service and information. Business and information are on the Y(+) and Y(-) axis while service is on the X axis.

As each component can either be categorized into b (functional) or i (non-functional information), either the b or i value needs to be zero.

Intensity = $(s^2 + b^2 + i^2)^{1/2}$ where s (service), b (business) and c (component) are all the results from group decision; the maximum value of intensity = $(1 + 1)^{1/2}$ which is close to 1.414.

In Figure 2, domain space is further partitioned into four zones; B and I remain the same while S is transformed into S-S form, the values of which are derived through parsing analysis. If the component is classified as a common component, the intensity should be plotted on the X(+) axis, otherwise the aggregated service will be plotted on the X(-) axis. Following is a description of these zones.

Tendency analysis is shown in Figure 3:

Let $\theta = \theta 1$ ($\theta 1, \theta 2, \theta 3, \theta 4$ associated with zone 1, 2, 3, 4 separately)

We derive θ by, $\theta = \sin^{-1}(s/\sqrt{(s^2+b^2)})$

if $\theta 1 < 45^0$ means that the tendency in C (commonbe component) is stronger than B

for the same reason, any of $\theta = \theta 2, \theta 3, \theta 4$, if $\theta < 45^0$, the tendency is on the X axis (either C or A).

A component's characteristics and its importance in a particular dimension can be the reference for implementation priority; more importantly, this produces a plotting map. When the skeleton of integration is determined, the map clearly tells us what service component needs to be the core within the skeleton.

The evaluation of intensity and tendency provides us with key information. A detailed evaluation follows:

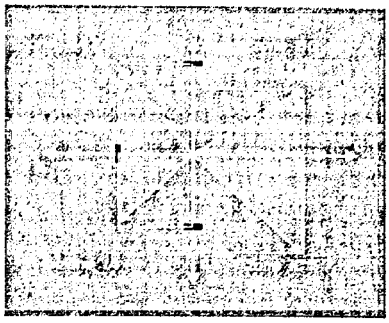


Figure 3. B-C-I-A assessment

To partition the space into four dimensions of evaluation, let
vector b denote the intensity of a service in Business dimension
vector c denote the intensity of a service in Common degree dimension
vector a denote the intensity of a service in Aggregation degree dimension
vector i denote the intensity of a service in Information dimension

To compute the intensity value of the vector Γ, let
vector $\Gamma 1$ denote the intensity in zone BC
vector $\Gamma 2$ denote the intensity in zone IC
vector $\Gamma 3$ denote the intensity in zone IA
vector $\Gamma 4$ denote the intensity in zone BA

$$\Gamma 1 = \frac{\sqrt{b^2+c^2}}{2} \quad \Gamma 2 = \frac{\sqrt{i^2+c^2}}{2} \quad \Gamma 3 = \frac{\sqrt{i^2+a^2}}{2} \quad \Gamma 4 = \frac{\sqrt{b^2+a^2}}{2}$$

To compute the intensity in the combination of dimension, let
vector $\theta 1$ denote the angle between vector $\Gamma 1$ and C in zone BC
vector $\theta 2$ denote the angle between vector $\Gamma 2$ and C in zone IC
vector $\theta 3$ denote the angle between vector $\Gamma 3$ and A in zone IA
vector $\theta 4$ denote the angle between vector $\Gamma 4$ and A in zone BA

$$\theta 1 = \sin^{-1}(b/\Gamma 1) \quad \theta 2 = \sin^{-1}(i/\Gamma 2) \quad \theta 3 = \sin^{-1}(i/\Gamma 3) \quad \theta 4 = \sin^{-1}(b/\Gamma 4)$$

In the combination scenario, let
vector $\Gamma 12$ denote the area covered by $\Gamma 1$ and $\Gamma 2$, and angle $(\theta 1 + \theta 2)$
vector $\Gamma 34$ denote the area covered by $\Gamma 3$ and $\Gamma 4$, and angle $(\theta 3 + \theta 4)$

$$\Gamma 12 = \frac{c(b+i)}{2} \quad \Gamma 34 = \frac{a(b+i)}{2} \quad \Gamma 23 = \frac{i(a+c)}{2} \quad \Gamma 14 = \frac{b(a+c)}{2}$$

In B-C-I-A partitions, the intensity is classified into three different levels, high (1 ~ 1.414), medium (0.5-1) and low (0 ~ 0.5). When we set up the priority for each service component, components within the same level of intensity classification need to have their tendency checked. The priority is as follows: BC > CI > BA > AI. The reason we set up this priority sequence is that the common component has implementation priority if both components fall within the same level. As most of our scenarios use the top-down approach, the priority in the business dimension is higher than that of an information (or non-functional) component. From the above evaluation, we derive the vector (intensity and tendency) of each service component within a project. The plotting map provides us with the direction of implementation priority and strategy, and also produces the information. Here we use the example of the plotting map in Figure 2 for deriving the implementation priority for each component.

Once the vertical skeleton and service extension are built, the interoperability reflected from the business layer has the capability of integrating cross-organization and cross-geographical constraints, as fully transparent services become possible if each system or organization follows an industry standard by using metadata strategy and an industry framework. An example of this would be a case in which several key industry vendors have built up an industry framework for service providers in that particular industry to follow, starting from business domain analysis and working through to component deployment. Each service provider would thus benefit from the industry's common services and the common interface, particularly when merger and acquisition cases (B2B, B2C) are involved. Any business activity becomes easier when the various parties involved have a common understanding and a consensus on business terms, service definitions, interfaces and component specifications, particularly if these become transparent and common practices. The important concept here is that once the metadata of business, service and information constructs an industry standard, the organization's vertical skeleton has the opportunity to integrate across multiple organizations for an easier global practice; this is a long-term industry objective which we call service virtualization in a fully transparent environment.

The semantic interpretation from business to service, and from service to technical need decomposition process to work on those selected business components, so each business component can be decomposed into business use cases for further breakdown. Service components derived from business use case are the mediation between business and technical element which integrate all business semantic, interfaces and technical configuration into a logical component, initially all the service components are in atomic level [6] [7].

The interoperability skeleton is an example of how the micro interoperability in the service layer, aligned with the component layer in micro mechanism, provisions the macro service in business. The foundation of this skeleton is

achieving interconnectivity between service components by using a common component and an aggregated component. We further develop the service virtualization on top of this interoperability skeleton which we call enterprise vertical integration, in extending the integration across industry and geographical areas. This is the virtualization service in a horizontal direction: once the industries of different regions build up their semantic interoperability in three metadata repositories (business, service and information), they can synchronize their business, operation and strategy.

Assume the value we derived from the political model $\Phi^*(SC^k_i)$ satisfies the eigenvalues approach and works as an eigenvalue of the rational model.

In n × n matrix, the rational model applys the approach of S-S in deriving common component and aggregation from the matrix of $\Psi(S^k_{ij})_{(1<i<n, 1<j<n)}$

In the equation $\Psi(S^k_{ij})\ X = \Phi^*(SC^k_i)\ X$, X denotes the eigenvector set; each set can be a pattern of common service. The implementation priority is based on the intensity of component reusability and frequency of fixed pattern. There is a possibility we may be able to use the above equation in deriving the patterns; however, how to best use a factor in balancing the left-hand side of the equation which concerns rational model parsing, and the right-hand-side political model derivation, is a concern which needs further development.

We use the terminology of "semantic heterogeneity" to denote different interpretation of same information. Semantic heterogeneity has been investigated in various literatures (Naiman Ouskel 1995 Ram Park 2004) [8] without a consensus. Goh characterized the distinction by structure (i.e. the logical organization of data) versus differences in interpretation (i.e., the meaning of data) whereas in the classification of Ram and Park (2004) [9], semantic heterogeneity exists in the data level as well in the schema level; while in Kashyap and Sheth (1996).

Based on the analysis of M. Selvage et al., once we have the priority of component implementation ready the roadmap is decided; at this point there are several different implementation patterns to achieve semantic interoperability:

- Point-to-point semantic integration
- Hub-and-spoke semantic integration
- Master data management (MDM) pattern
- Industry information model
- The semantic web

The scope can include business units, enterprises and groups of enterprises within the same industry or cross-industry. The bigger the circle of semantic interoperability is, the more reusable the results will be in the long run but also the more difficulty there will be in the areas of coordinating and reaching consensus. Moreover, an individual company will have less control over tailoring the reusable assets to its unique business needs and determining the desirable features. For these reasons, some companies may use internal proprietary data models for their internal integration efforts and industry standard models for B2B. Enterprises should take a balanced view in considering the tradeoffs of adopting different patterns of semantic interoperability [10].

Conclusion

We repeat the political model in a use case deconstruction to derive service components and physical elements. The group members validate both service components and component elements by using the same method with differently weighted roles. The evaluation between service components and physical components is based on how significant their contribution to the business goal is from a logical and physical information component perspective; a higher degree of information aggregation of C within B-S-C gives a higher preference weight to that physical component. The multiplication of B and S significant factors represents the significance of S within the K-B-S-C transformation; using the same method, the multiplication of B and C significant factors represents the significance of C within the transformation.

Each project or enterprise activity has its unique strategy, business context or KPI that first needs to be converted to business components. Once the common component and aggregation services are identified and developed, the alignment can be reused in a top-down K-B-S-C approach by selecting the same business components, service components and physical components to establish the business goal; however, in most project scenarios only part of the empirical K-B-S-C transformation process can be reused. This leaves a large portion of the alignment as either new material or a deep modification which requires starting from business component selection. In the service-component-based architecture, interoperability within K-B-S-C is the key in building the skeleton of enterprise integration in a

vertical direction. To cope with this objective, we apply both a political model and a rational model to validate the components in each layer. The political model applies group decision-making while the rational model applies parsing analysis; both models work on component validation with different focus of scope.

We apply B-C-I-A assessment in evaluating the component's characteristics. Based on this analysis, we produce visibility for the tendency and intensity of B-C-I-A domains; we also ascertain the level of common access and aggregation dependency for further development. The metadata information which is stored in repository provides the priority and dependency in our implementation. In the K-B-S-C model analysis, a bottom-up approach using rational model parsing is applied to S-S transformation. This step is the last one in K-B-S-C transformation and the objective is to derive common components and service aggregation. The statistics of parsing results construct a matrix of common components and aggregated coarse-grained components which can be candidates for aggregation service and end-to-end process; this bottom-up approach attempts to create visibility of common services in either component units, interface or composite components, along with their priority and dependency. This provides the architects with a roadmap for building up their enterprise integration and strengthening the alignment and interoperability within the K-B-S-C transformation.

References

1. Mishra A., Ripley M., and Gupta A. (2002), "Using Structural Analysis to Mediate XML Semantic Interoperability", MIT Sloan Working PP 4345-02 No. 1024-05 February 2002
2. Moulton A., Madnick M., and Siegel M. (2003), "Context Interchange Mediation for Semantic Interoperability and Dynamic Integration of Autonomous Information Sources in the Fixed Income Securities Industry", No 4404-02 Working papers from MIT Sloan School of Management
3. Cambrin D., Demeule L., Dumedah G., Hoare D. And Simmonds K. (2004), "Sub-Surface Data Management", Geography 455 Advanced Issues in GIS March 2004
4. L. Yu, (2005), "Categorization of Common Coupling in Kernel-Based Software," 43rd ACM Southeast Conference, 2005
5. J. Hunter , (2001), "MetaNet - A Metadata Term to Enable Semantic Interoperability Between Metadata Domains", Journal of Digital Information, Volume 1 Issue 8
6. M. Staab, N. Stojanovic, R. Studer, and Y. Sure, (2001), "Semantic Portal - The SEAL approach. to appear in: Creating the Semantic Web.", MIT Press
7. O. Valkeapää and E. Hyvönen, (2006), "Semantic Annotation with Browser-based Annotation Tool", 1st Asian Semantic Web Conference
8. Kashyap V. and Sheth A. (1996), "Semantic and Schematic Similarities between Database Objects: A Context-based approach", VLDB Journal: Very Large Data Bases
9. Park J. and Ram S. (2004) "Information systems interoperability: What lies beneath?", ACM Trans. Inf. Syst. 22(4) 595-632
10. M. Selvage, D. Wolfson, B. Zurek, E. Kahan, (2006), "Achieve semantic interoperability in a SOA", IBM Academy 2006

T5: Micro Process Design and Services Virtualization

Publication: Southeast07 (IEEE) Conference Procedeing
22 -25 March 2007, Richmond, VA, USA (http://www.southeastcon.org/2007)
ISBN: 1-4244-1029-0
Publication Contact: Professor Gang Tao
Technical Conference Chair
Dept. of E.E., University of Virginia
gt9s@ece.Virginia.edu

IEEE SoutheastCon 2007

Date: Tue, 23 Jan 2007 13:17:30 -0800
From: gt9s@ece.Virginia.EDU Add to Address Book Add Mobile Alert
To: rwu_academic@yahoo.com
CC: gt9s@ece.Virginia.EDU
Subject: Your SoutheastCon 2007 Submission (Number 98)

Dear Mr. Raymond Wu:

On behalf of the SoutheastCon 2007 Technical Program Committee, I am pleased to inform you that the following submission has been accepted to the 2007 IEEE Southeast Conference:

Micro Process Design and Services Virtualization in Enterprise Integration

as a full paper.

Reviewer's Scores

Originality: 5
Technical strength: 4
Presentation: 5
Overall evaluation: 4

Comments

It is a new concept for me to learn. It is very interesting.
Abstract is clear and to the point. The Key idea is brought out very clearly.
All concepts are clear.

Micro Process Design and Services Virtualization

Raymond Wu
IBM Software Group
raymond@de.ibm.com

Abstract

In this research paper we deal with the service virtualization concept based on the common component concept we implemented in previous research and on the component-based design foundation we build in the current research.

Introduction

The importance of determining the common component and service aggregation is in building up an enterprise skeleton of vertical interoperability between business, service and information. This task is crucial, as in the business domain analysis stage, the impact area and scope to business components are minimized in decision making. Additionally, the business component is deconstructed into service components and physical elements for top-down alignment while the common component is not simply a symbolic service but instead requires a bottom-up approach to work on common areas; this serves to fine tune the common components and interfaces from the perspective of enterprise integrity and performance. The service aggregation is also important as the top-down deconstruction followed by bottom-up composition forms the skeleton of vertical interoperability and potential component-based implementation of the service process.

The string $\Phi(r_n, r_m) = \Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m$ is a path from r_n to r_m, we call the "service route" and $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$ a "composite service" between r_{i-1} and r_{i+1}, we use the symbol $\Phi_{i, i+1}$ to denote a composite service.

When frequency $F(\Phi_{i, i+1}) = F(\Phi_i \times \Phi_{i+1})$ is higher than threshold σ, we aggregate the concatenated alphabet (part of the string) $\Phi_i \times \Phi_{i+1}$ into $\Phi_{i, i+1}$, this implies that the original state r_i between r_{i-1} and r_{i+1} will no longer exist for this composite service.

Let M be an NFA so that $M = (S, \Sigma, T, s_0, A)$, and Φ be a string over the alphabet Σ. M accepts the string Φ if there exists both a representation of Φ of the form $\Phi_1, \Phi_2 \ldots \Phi n$, $\Phi i \in (\Sigma \cup \{\varepsilon\})$, and a sequence of states $r_0, r_1, \ldots, r_n$, $r_i \in S$, thus meeting the following conditions:

1. $r_0 \in s_0$
2. $r_i \in T(r_{i-1}, \Phi i)$, for $i = 1, \ldots, n$
3. $r_n \in a_t$

$M = (S, \Sigma, T, s_0, A)$ where $\Sigma = \{\Phi i\}$ for $i = 1,2 \ldots, n$

$S = \{ S_{ij} \}$ for $i = 1 \ldots 6$, and $j = 1 \ldots 3$ => the union of state

$\Sigma = \{ s_0, s_1, s_2, s_3, s_4, s_5 \}$ => the service component is the alphabet for reading

$r_0 = \{ S_{11}, S_{42}, S_{61} \}$ => the initial state

$r_i \in T(r_{i-1}, \Phi i)$, for $i = 1, \ldots, n$ => the sequence state

$r_n = \{ S_{33}, S_{51}, S_{63} \}$ => the final state

$E = \{ \varepsilon i \}$ for $i = 1, \ldots, n$, we replace $T(r_{i-1}, \Phi i)$ by 'εi' which represents a semantic event-driving state from r_{i-1} to r_i by using service Φi mediation.

Let S^* denote the union of the concatenation of S with a frequency higher than 1, and $F(\Phi)$ denote the frequency of string Φ in the parsing process, and we assume component element c_i associated with service component s_i

When the use case is decomposed into service and component elements in atomic form, we derive the following scenarios:

Semantic event aggregator

1.At the very beginning of a use case, the first event triggers service Φ_1

$\varepsilon^1 : r_0 \times \Phi_1 \rightarrow r_1$

2. When the service event Φi ($i <> 1$) is read and triggers the service component, we express the state transformation as follows:

$\varepsilon^i : r_{i-1} \times \Phi_i \rightarrow r_i$, we further derive the transformation as follows:

$\varepsilon^{i+1} = (r_i) \times \Phi_{i+1} \rightarrow r_{i+1} = (r_{i-1} \times \Phi_i) \times \Phi_{i+1} \rightarrow r_{i+1}$

which is in equilibrium to $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$, and derives $\varepsilon^{sm}: r_n \times (\Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m) \rightarrow r_{m\,(n<m)}$, we call this the "semantic event aggregator."

3. As the final step of a use case, the final event trigger service Φ_n

$\varepsilon^n : r_{n-1} \times \Phi_n \rightarrow r_n$, Φ_n is a terminator, an example of which is the settlement of a daily transaction.

In any component context a formal interface contract for the invocation syntax is important. The semantics issue (the meaning of parameters and so forth) has to be solved as well (domain modeling). This is crucial to any business-to-business (B2B) and dynamic invocation scenario. Such scenarios are cornerstones of the service component vision

1-4244-1029-0/07/$25.00 ©2007 IEEE.

of being flexible and agile in response to the new business needs in a world of mergers and acquisitions, business transformation, globalization and other relevant factors. [9]

Main route

When the business architect checks the result from semantic event aggregator - ε^m which is more an upper-layer use case scenario, service route - $\Phi_{i,\ i+1}$ which is in the service layer and its frequency - $F(\Phi_{i,\ i+1})$, if $F(\Phi_{i,\ i+1})$ is much higher than threshold σ, then from the business perspective the event aggregator ε^m either consists of many composite services or is business critical. This means the aggregation service is the key within business and is crucial in daily operation from service perspectives and that fine tuning is feasible from a multi-fine grained boundary to a coarse grained service. We call this the "main route" of the enterprise skeleton; in other words this is the highway between A and B. Virtual coarse grained service eliminates the boundary and leverages many fine-grained components which can be easily configured by adding or modifying the component dynamically to achieve the broader coverage of main route services. For this reason main route is one of crucial services in enterprise integration. One example is in the Telecommunication industry that incorporates an open service account, an open billing account and provisioning options for a new customer. This constitutes a main route for "new customer service" coarse grained service, though it is aligned with fine-grained components to support customer preference, service criteria and billing options which are branches within this main route.

Alternative service shortcut

In service route ε^{nm}: $r_n \times (\Phi_{n+1} \times \Phi_{n+2} \ldots\ldots \times \Phi_m) \rightarrow r_{m,\ (n<m)}$ if these exist in a composite service Φ_i that r_i is omitted ($n < i < m$). All the above terminology is shown in Figure 1 regarding the formation of composite, alternatives and main route. All of these critical options are enabled by the automata model as expressed in Figure 2.

Common interface

Common interface is also a means to achieve common components. S. Melnik proposes a universal interface which avoids common models and languages; this is the Generic Interoperability Framework which was developed to facilitate the integration of heterogeneous information systems. Within this framework individual communication protocols, data manipulation languages and data are represented in a generic manner while preserving their ontological variety; this process indicates how a mediation infrastructure can be built using canonical wrappers, operational requirements of integration and layered architecture for component design [11].

Enterprise integration

The iteration through each use case will be tracked for its path and service statistics so that once we parse all use cases we should be able to see the frequency of use of each service as well as the path and dependency between services. Where there is a high frequency of services we use the term "common services" and focus on the potential for reuse and minimization of the integration interface. In the case of service paths in which several services have a common sequence and appear in use cases a certain number of times past the threshold, the services can be aggregated to a composite service or potential process. This needs to be designed at an enterprise level so that each organization has the same concept of and specification for this common service. Service aggregation means a composite component consisting of at least two components that form a unique sequence of execution order with a common interface within their boundaries; this composite component will be treated as a coarse grained service in provisioning the upward aggregation of services.

Service virtualization

Once the vertical skeleton and service extension are built, the interoperability reflected from the business layer has the capability of integrating cross-organization and cross-geographical constraints, as fully transparent services become possible if each system or organization follows an industry standard by using metadata strategy and an industry framework. An example of this would be a case in which several key industry vendors have built up an industry framework for service providers in that particular industry to follow, starting from business domain analysis and working through to component deployment. Each service provider would thus benefit from the industry common services and the common interface, particularly those involving merger and acquire cases (B2B, B2C). Any business activity becomes easier when the various parties involved have a common understanding and a consensus on business terms, service definitions, interfaces and component specification, particularly if these become transparent and common practices. The important concept here is that once the metadata of business, service and information constructs an industry standard, the organization's vertical skeleton has the opportunity to integrate across multiple organizations for an easier global practice; this is a long-term industry objective which we call service virtualization in a fully transparent environment.

The interoperability skeleton shown in Figure 3 is an example of how the micro interoperability in the service layer, aligned with the component layer in micro mechanism, provisions the macro service in business. The foundation of this skeleton is achieving interconnectivity between service components by using a common component and an aggregated component. We further develop the service virtualization on top of this interoperability skeleton, which we call enterprise vertical integration, in extending the

1-4244-1029-0/07/$25.00 ©2007 IEEE.

integration across industry and geographical areas. This is the virtualization service in a horizontal direction; once the industries of different regions build up their semantic interoperability in three metadata repositories (business, service and information), they can synchronize their business, operation and strategy even if they have their local implementation extended from the interoperability skeleton. Figure 4 demonstrates the resolution so that the limitation of business, geographical area, products, application, and data structure become transparent. This is what we call service virtualization; we predict that virtualization of services will become a priority in industry as semantic interoperability is in high demand to solve the issues in both vertical interoperability and horizontal interconnectivity.

Conclusion

In transformation of service aggregation we explored the possibility of applying automata theory in service aggregation. In enterprise service provisioning and virtualization, we constructed the skeleton of enterprise service with leverage to each layer and the interoperability of the layers. Service virtualization is a process of leveraging enterprise integration strategy within a specific geographical region or across regions. This can be an industry standard scenario such as a credit card transaction in which the service, geography and systems are fully transparent to both cardholders and merchants. A full virtualization environment needs to be built on top of the foundation of component design and service transformation process in order to apply the vertical enterprise integration approach to this final stage which is the goal of our research work. The standardization of metadata technology and common services and components are all requirements in this stage.

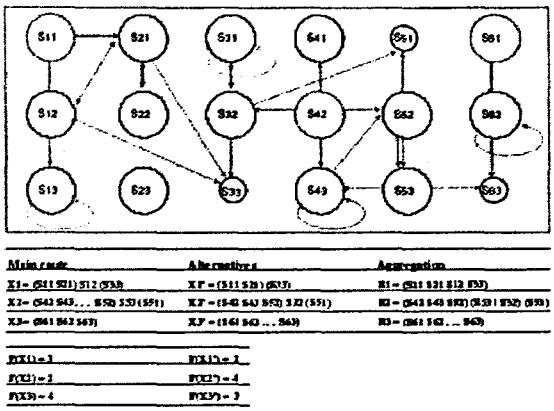


Figure 1. From aggregation to Main route

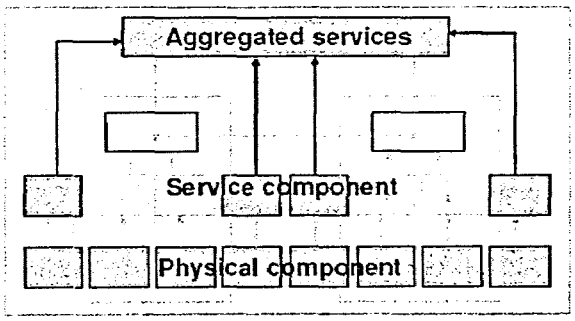


Figure 2. Aggregated service in common services

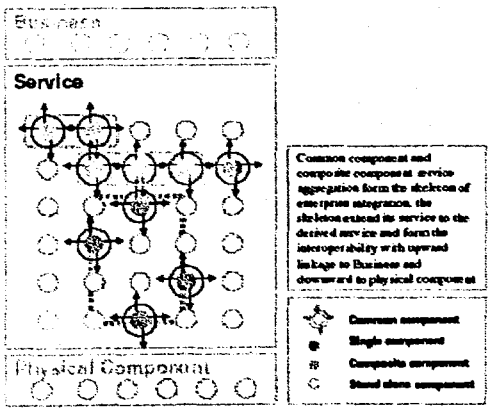


Figure 3. Skeleton of enterprise integration interoperability

Reference

1. DSouza,(2001)," Model-Driven Architecture and Integration Opportunities and Challenges", Kinetium 2001
2. Philippe Kruchten, (1998), "Modeling Component Systems with the Unified Modeling Language", 1998 International Workshop on Component-Based Software Engineering
3. Jie Lu, Guangquan Zhang & Fengjie Wu, (2005), "Web-based Multi-Criteria Group Decision Support System with Linguistic Term Processing Function", IEEE Intelligent Informatics Bulletin June 2005 Vol.5 No.1
4. John F. Sowa, (2000), "Ontology, Metadata, and Semiotics", ICCS 2000 AI #1867 Springer-Verlag pp. 55-81
5. D. Oberle, S. Staab and A. Eberhart, (2006), "Semantic Management of Distributed Web Applications", IEEE
6. R. Barcia and J. Brent, (2005), "Building SOA solutions with the Service Component Architecture", IBM Research
7. Raymond Wu., (2006), "Enterprise Integration in semantic metadata", IEEE (IT professional) ITPro-0055-0806
8. L. de Alfaro and T. A. Henzinger, "Interface theories for component-based design." in Distributed Systems Online, vol. 7, no. 4, 2006, art. no. 0605-o5001 Proceedings of theFirst International Workshop on Embedded Software (EMSOFT), Lecture Notes in Computer Science 2211,Springer-Verlag, 2001, pp. 148-165.
9. O. Zimmermann, P. Krogdahl, C. Gee, (2004), "Elements of Service-Oriented Analysis and Design", IBM Academy
10. E. Gramond and S. H. Rodger, (1999), "Using JFLAP to Interact with Theorems in Automata Theory", Thirtieth SIGCSE Technical Symposium on Computer Science Education, ACM Press 1999 page 336-340
11. Sergey Melnik, (1999), "Generic Interoperability Framework", Distributed and Parallel Databases Journal 1999
12. B. Hutchison and M. Schmidt and D. Wolfson, (2005), "SOA programming model for implementing Web services", IBM Research 2005
13. D Roman et al. (2005), "Web Service Modeling Ontology (WSMO)", European Commission under the projects DIP 2005

1-4244-1029-0/07/$25.00 ©2007 IEEE.

T6

Metadata Computing in CRM systems

Technical White Paper version 2.0 September, 2004

ORACLE

Raymond Wu (ext.3194) Principle Consulting Architect, Oracle Corporation

(Presentation to EBT Telecom, Taiwan) For GC Internal Distribution

Objective:

This program try to synchronize the source data change to remote sites by using metadata broadcasting. Source site, in this case is OES/CMS and Remote sites are OCR (MS SQL-server case) and CWS (Oracle/Unix case), both cases can be expanded to unlimited heterogeneous environments, it can be in different database or format from source site, it guarantee single source of truth, and one point change trig each remote site to commit automatically.

Background:

Currently VB program from MS-server front end takes the data change of Promotion and Rating plan to run against OES/CMS, and manually type the data and send it to remote to apply the change against their database (CWS & OCR), the manual work starting from VB data input will be automated in this program that once administrator finish their data input, a click on the keyboard will generating the change from OES/CMS to remote format (CWS & OCR), that means the *"pr_gen"* engine use the schema mapping (between source and remote) to transform the data to remote DB format and create the commands to 'fly' the sql/shell script to remote and execute it.

The whole process need no effort at remote sites and if required, it can be upgraded to 'staging DB' in the future to allow the administer to validate the data input before the data goes to all (source and remote) sites. The only thing need to maintain from time to time is when the schema (at any site) is changed, *pr_mapping* need be run to re-generating the mapping.

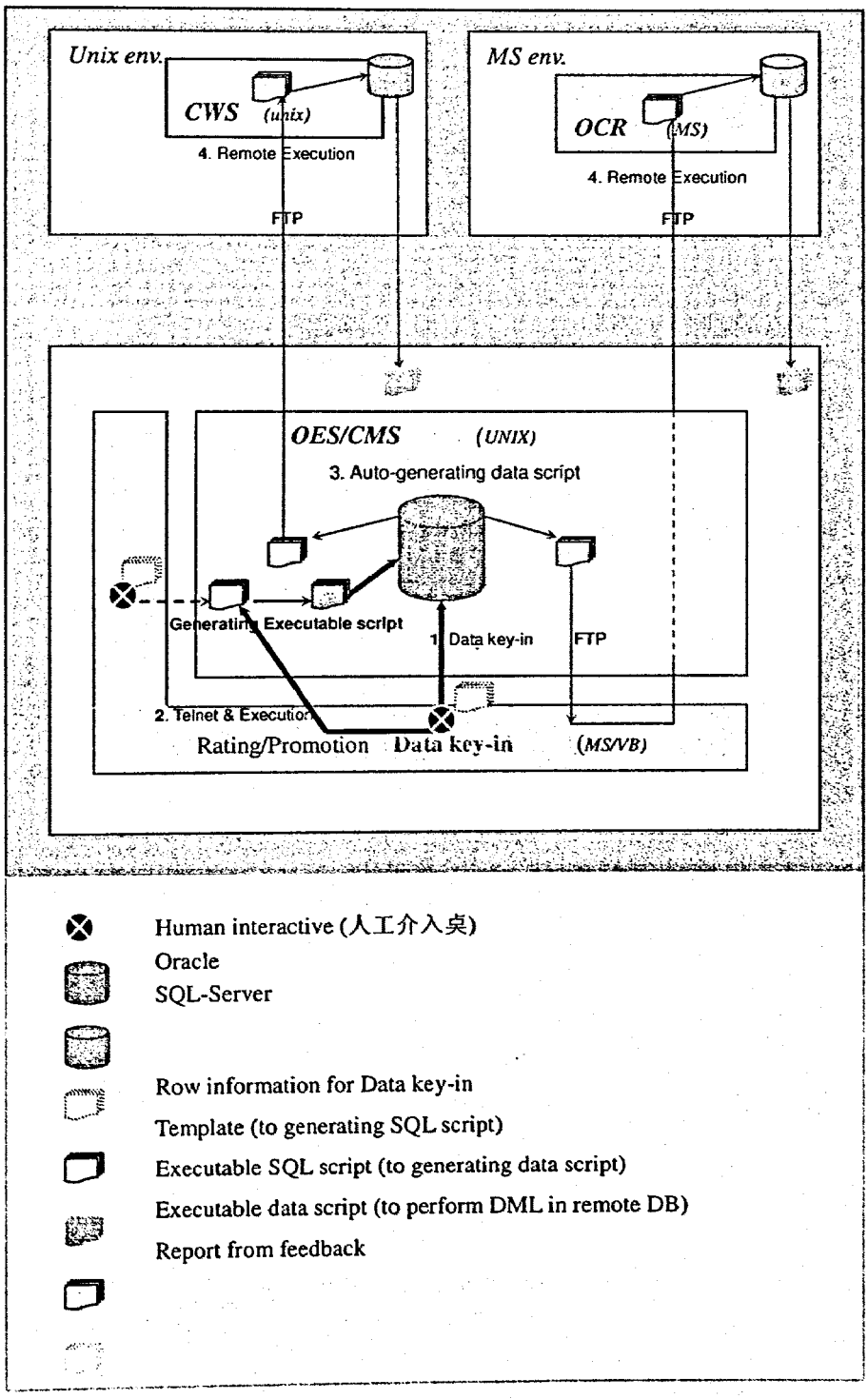
The synchronization mechanism use this mapping to consolidate the extracted data with the generatingd sql/shell scripts into *"pr_run_file"* and ftp it to the remote site, the pr_run_file has 2 different formats (Unix for Oracle and MS for SQL-server), depend on the parameter (ocr or cws) given, it will be run at remote site (*rsh* for Unix and *osql* for MS), the whole process takes around 5 seconds, however remote data change take no time from old data to new data.

This program is consisted of 2 major parts;

1. pr_gen (the engine to apply the change from source to remote), run "pr_gen ocr" (or cws) from VB
2. pr_mapping (generating mapping between source and remote schema) run "pr_mapping ocr" (or cws) from VB
3. The limitation of this program is, the data size better to be small (i.e.; less than 10k records). Both of the cursor engines are introduced as followed:

Promotion/Rating Real-time Data Synchronization
(即時資料同步作業)

- 以 Broadcasting 方式進行一對多的 Data Synchronization, OES/CMS 為唯一允許變更資訊的 Data Source Input
- Sync 時機是在 OES/CMS 完成 Data key-in, 在最後按下 'Commit' 後以 FTP 及 Remote Shell Execution/ OSQL 一次完成, 目的在避免重複人工 key-in 作業, 以及錯誤率. Remote 端目前分 Unix (CWS) 及 MS (OCR) 二大環境
- 人工作業僅限 OES/CMS 上的 data key-in 及發生頻率極低的 schema change, 其餘皆以 Shell/SQL 自動化處理



Advanced SQL cursor-recursive computing

```
CURSOR c01 IS SELECT lt_name from promotion_data_mapping_level where upper(system)=upper(SYSTEM_NAME) order by -lt_level;
CURSOR c02 IS SELECT lt_name from promotion_data_mapping_level where upper(system)=upper(SYSTEM_NAME) order by lt_level;
CURSOR c1 IS SELECT distinct lt_name, rt_name from promotion_data_mapping where upper(system)=upper(SYSTEM_NAME) and
lt_name=lt_name0;
BEGIN
lf_name_str :='';
rf_name_str :='';
                OPEN c02;
                LOOP
                FETCH c02 INTO lt_name0;
                EXIT WHEN c02%NOTFOUND;
                OPEN c1;
                LOOP
                FETCH c1 INTO lt_name1, rt_name1;
                EXIT WHEN c1%NOTFOUND;
                                rf_name_str :='';
                                lf_name_str :='';
                                rpf_str :='';
                                lpf_str :='';
                                rf_str :='';
                                lf_str :='';
                                OPEN c2;
                                LOOP
                                FETCH c2 INTO lf_name2, rf_name2, lf_type2, rf_type2, lpf_name, rpf_name;
                                EXIT WHEN c2%NOTFOUND;
                                lpf_str := lpf_str||','||lpf_name;
                                rpf_str := rpf_str||','||rpf_name;
                                lf_str := lf_str||','||lf_name2;
                                rf_str := rf_str||','||rf_name2;
                                lf_name2 := ''''||lf_name2||'''';
                                If substr(upper(lf_type2),1,3) not in ( 'NUM', 'INT' ) then
                                lf_name2 := ''''||lf_name2||'''';
                                end if;
                                If substr(rpf_str,1,1) = ',' then
                                rpf_str := substr(rpf_str,2,length(rpf_str)-1);
                                end if;
                                end LOOP;
                                str1 := substr('select "insert into '||rt_name1||'_v ('||rf_name_str||') values ('||lf_name_str||');" from
                                '||lt_name1||'_v;',1,200);
                                str2 := substr('select "insert into '||rt_name1||'_v ('||rf_name_str||') values ('||lf_name_str||');" from
                                '||lt_name1||'_v;',201,200);
                                dbms_output.put_line ('STR1='||str1||'');
                                dbms_output.put_line ('STR2='||str2||'');
                                dbms_output.put_line ('STR_T=');
                                dbms_output.put_line ('echo2');
                                close c2;
                end LOOP;
                close c1;
                end LOOP;
close c02;
OPEN c01;
LOOP
FETCH c01 INTO lt_name2;
EXIT WHEN c01%NOTFOUND;
                OPEN c03;
                LOOP
                FETCH c03 INTO lt_name3, rt_name3;
                EXIT WHEN c03%NOTFOUND;
                dbms_output.put_line ('drop table '||rt_name3||'_bk;');
                dbms_output.put_line ('create table '||rt_name3||'_bk as select * from '||rt_name3 ||';');
                dbms_output.put_line ('delete from '||rt_name3||';');
                end LOOP;
                close c03;
end LOOP;
close c01;
EXCEPTION
WHEN NO_DATA_FOUND THEN
Rollback;
dbms_output.put_line ('NO_DATA_FOUND');
WHEN OTHERS THEN    -- handles all other errors
ROLLBACK;
dbms_output.put_line ('Other Error ');
End;
/
spool pr_gen.tmp1
exec pr_gen ($DB_EXE);
spool off;
EOF
echo > rm_gen.sql
cat pr_gen.tmp3|sed 's/STR_T=/STR_T=$STR1$STR2$STR3$STR4/g'>              pr_gen.tmp4
cat pr_gen.tmp4|sed 's/     //g' >          pr_gen.tmp5
cat pr_gen.tmp5|sed 's/echo0/echo $STR_T1/g'     >              pr_gen.tmp6
```

```
cat pr_gen.tmp11 |grep -v "create table"    >                pr_gen.tmp12
cat pr_gen.tmp12 |grep -v STR_T2                              >                pr_gen.exe1
cat pr_gen.tmp12 > rmt_gen.exe1
cat rmt_gen.exe1 |sed   's/$STR_T2/$STR_T2   >>   rmt_gen1.sql/g' > rmt_gen1.exe
chmod 700 rmt_gen1.exe
echo > rmt_gen1.sql
rmt_gen1.exe
pr_gen.exe1 > pr_gen1.sql
$ORACLE_HOME/bin/sqlplus $LOCAL_UID/$LOCAL_PWD << EOF
set linesize 400
prompt off;
set pagesize 999;
set heading off;
set echo off;
set termout off;
set serveroutput ON size 1000000;
spool                                                rmt_gen2.sql
@pr_gen1.sql
spool off;
EOF
echo export ORACLE_SID   >> rmt_gen
echo NLS_LANG=AMERICAN_AMERICA.ZHT16BIG5   >> rmt_gen
echo export NLS_LANG   >> rmt_gen
echo $REMOTE_ORACLE_HOME'/bin/sqlplus ' $REMOTE_ORACLE_UID/$REMOTE_ORACLE_PWD' << EOF'    >> rmt_gen
echo spool $REMOTE_LOG_FILE   >>            rmt_gen
cat pr_gen.tmp1 |grep 'drop table' >>                rmt_gen
cat pr_gen.tmp1 |grep 'create table' >>      rmt_gen
cat pr_gen.tmp1 |grep 'delete table' >>              rmt_gen
cat rmt_gen1.sql                    >>       rmt_gen
cat rmt_gen2.sql|grep 'insert' >>   rmt_gen
echo spool off              >>                rmt_gen
echo commit;                >>                rmt_gen
echo EOF >>                                   rmt_gen
chk=1
while [ `expr $chk` -gt 0 ]
do
cat rmt_gen|grep ,, > rmt_gen.log
cat rmt_gen |sed 's/,/,null,/g' > rmt_gen.tmp
sleep 2
chk=`wc -l rmt_gen.log|awk '{print "",$1}'`
echo $chk
cat rmt_gen.tmp > rmt_gen
done
mv rmt_gen $LOCAL_RUN_FILE
chmod 700 $LOCAL_RUN_FILE
rm $LOCAL_LOG_FILE
ftp -n -v $REMOTE_HOST_NAME <<   EOF
user $REMOTE_UID $REMOTE_PWD
put   $LOCAL_RUN_FILE   $REMOTE_RUN_FILE
disconnect
!rsh -l $REMOTE_UID $REMOTE_HOST_NAME chmod 700 $REMOTE_RUN_FILE
!rsh -l $REMOTE_UID $REMOTE_HOST_NAME $REMOTE_RUN_FILE
quit
EOF
ftp -n -v $REMOTE_HOST_NAME <<   EOF
user $REMOTE_UID $REMOTE_PWD
get   $REMOTE_LOG_FILE   $LOCAL_LOG_FILE
quit
EOF
fi
if [ $CASE = 'MS' ]
then
CREATE or replace PROCEDURE pr_gen (SYSTEM_NAME in char)   is
CURSOR c01 IS
SELECT lt_name from promotion_data_mapping_level where upper(system)=upper(SYSTEM_NAME) order by -lt_level;
CURSOR c02 IS
SELECT lt_name from promotion_data_mapping_level where upper(system)=upper(SYSTEM_NAME) order by lt_level;
CURSOR c03 IS
SELECT distinct lt_name, rt_name from promotion_data_mapping where upper(system)=upper(SYSTEM_NAME) and lt_name=lt_name2;
CURSOR c1 IS
SELECT distinct lt_name, rt_name from promotion_data_mapping where upper(system)=upper(SYSTEM_NAME) and lt_name=lt_name0;
CURSOR c2 IS
SELECT distinct lf_name, rf_name, lf_type, rf_type, lpf_name, rpf_name from promotion_data_mapping where lt_name=lt_name1 and
rt_name=rt_name1 and upper(system)=upper(SYSTEM_NAME) and rf_name is not null;
BEGIN
lf_name_str :='';
rf_name_str :='';
                OPEN c02;
                LOOP
                FETCH c02 INTO lt_name0;
                EXIT WHEN c02%NOTFOUND;
                OPEN c1;
                LOOP
                FETCH c1 INTO lt_name1, rt_name1;
                EXIT WHEN c1%NOTFOUND;
                                rf_name_str :='';
```

```
                                lf_name_str :='';
                                rpf_str :='';
                                lpf_str :='';
                                rf_str :='';
                                lf_str :='';
                                OPEN c2;
                                LOOP
                                FETCH c2 INTO lf_name2, rf_name2, lf_type2, rf_type2, lpf_name, rpf_name;
                                EXIT WHEN c2%NOTFOUND;
                                If substr(lpf_str,1,1) = ',' then
                                lpf_str := substr(lpf_str,2,length(lpf_str)-1);
                                end if;
                                If substr(rpf_str,1,1) = ',' then
                                rpf_str := substr(rpf_str,2,length(rpf_str)-1);
                                end if;
                                If substr(lf_str,1,1) = ',' then
                                lf_str := substr(lf_str,2,length(lf_str)-1);
                                end if;
                                If substr(rf_str,1,1) = ',' then
                                rf_str := substr(rf_str,2,length(rf_str)-1);
                                end if;
                                end LOOP;
                                str3 := substr('select "insert into 'ilrt_name1ll'_v ('ilrf_name_strll') values ('ll lf_name_strll')" from
                                'illt_name1ll'_v;',401,200);
                                str4 := substr('select "insert into 'ilrt_name1ll'_v ('ilrf_name_strll') values ('ll lf_name_strll')" from
                                'illt_name1ll'_v;',601,200);
                        dbms_output.put_line ('STR1="drop view ' ilrt_name1ll '_v"');
                        dbms_output.put_line ('STR_T2=');
                        dbms_output.put_line ('echo1');
                        dbms_output.put_line ('STR1="create view ' ilrt_name1ll '_v (' ilrf_strll') as select "');
                        dbms_output.put_line ('STR2="'ilrpf_strll' from ' ilrt_name1ll' "');
                        dbms_output.put_line ('STR_T2=');
                            close c2;
                            lf_name_strll'); " from 'illt_name1);
                end LOOP;
                close c1;
                end LOOP;
close c02;
OPEN c01;
LOOP
FETCH c01 INTO lt_name2;
EXIT WHEN c01%NOTFOUND;
                OPEN c03;
                LOOP
                FETCH c03 INTO lt_name3, rt_name3;
                EXIT WHEN c03%NOTFOUND;
                dbms_output.put_line ('MS-command:drop table 'll rt_name3ll'_bk' );
                dbms_output.put_line ('MS-command:go' );
                dbms_output.put_line ('MS-command:select * into 'll rt_name3ll'_bk from 'll rt_name3);
                dbms_output.put_line ('MS-command:go' );
                dbms_output.put_line ('MS-command:delete from 'll rt_name3);
                dbms_output.put_line ('MS-command:go' );
                end LOOP;
                close c03;
end LOOP;
close c01;


EXCEPTION
WHEN NO_DATA_FOUND THEN
Rollback;
dbms_output.put_line ('NO_DATA_FOUND' );
WHEN OTHERS THEN    -- handles all other errors
ROLLBACK;
dbms_output.put_line ('Other Error ' );
End;
if [ $SYSTEM_NAME = 'CWS' ] || [ $SYSTEM_NAME = 'cws' ]
then
cp cws1.tmp 1.tmp
cp cws2.tmp 2.tmp
cp cws3.tmp 3.tmp
for file in `echo '1.tmp' '2.tmp' '3.tmp' '4.tmp' '5.tmp' '6.tmp'`
do
cp $file mapping_file
header=`head -1 mapping_file`
for line in `cat mapping_file |grep -v insert|sed 's/ //g' `
do
line=`echo $line"";"|sed "s/./:/g"`
line=`echo $line|sed "s/values (/values ('/g"`
line=`echo $line|sed "s/not/ not /g"`
line=`echo $line|sed "s/NOT/ NOT /g"`
line=`echo $line|sed "s/' //g"`
echo $header "'"$line >> mapping_file.sql
done
done
$ORACLE_HOME/bin/sqlplus $LOCAL_UID/$LOCAL_PWD << EOF
```

```
delete from promotion_data_mapping where upper(system)='CWS';
@mapping_file.sql
drop sequence seq1;
create sequence seq1;
update promotion_data_mapping set lf_name='f'||seq1.nextval where system='CWS' and lt_name= 'PROMOTION_GROUP';
drop sequence seq1;
create sequence seq1;
update promotion_data_mapping set lf_name='f'||seq1.nextval where system='CWS' and lt_name= 'PROMOTION_DATA';
drop sequence seq1;
EOF
fi
```

Preliminary DDL recursive meta-computing

(ETL_CR_PAR)

```
clear
echo
sqlplus ictdevb/ictdevb << EOF
spool etl_add_par.tmp;
@etl_add_par.sql
spool off;
EOF
cat etl_add_par.tmp|grep alter > etl_add_par.exe
sqlplus ictdevb/ictdevb << EOF
@etl_add_par.exe
delete from op_mgt;
insert into op_mgt(OPERATOR,PARTITION_NAME) select substr(partition_name,3,length(partition_name)-10), partition_name from dba_tab_partitions where table_name = 'COMPONENT_TABLE';
EOF
```

(etl_add_par.sql)

```
select 'alter table component_table split partition max_p at ('''||substr(t.table_name,18,7)||''''||','''||o.operator||''') into (partition P_'||o.operator||'_'||substr(t.table_name,18,7)||', partition max_p);' from dba_tables t, op_list o where table_name like 'STORED_COMPONENT%_01_1' and lower(o.status)='s' and
not exists (select 1 from op_mgt op where 'P_'||o.operator||'_'||substr(t.table_name,18,7)=partition_name) order by t.table_name,o.operator;
```

Extract Data from selected months

(ETL_DB_EXT)

```
clear
# Menu for selection
echo 'Please Enter Month <yyyy_mm> for Extract step in ETL process => '
echo 'Start Month: \c'
read start_month
echo 'End Month:   \c'
read end_month
start_date='stored_component_'$start_month'_01_1'
end_date='stored_component_'$end_month'_31_1'
start_date="'$start_date'"
end_date="'$end_date'"
echo > etl_date.exe
chmod 700 etl_date.exe
echo 'Please Wait for system to extract ETL Table ..... '
```

T7

Event Driven Batch Process Time-Volume-Condition logical controller

ORACLE

Presentation to EBT Telecom, Taiwan

Version 1.0
August 2004

Raymond Wu
Oracle Principal consulting architect

Objective

Event Driven Time-Volume-Condition Controller (EDTVCC) is an invention works on those events that doesn't need prompt response however timing is critical between event submitted and either transaction committed or query retrieved. The solution use metadata control and optimization between volume, time and status condition to improve the transaction integrity and performance.

Business scenarios of EDTVCC are such as Retailer, when facing customer's purchasing, the item of transaction initiate the event to store the transaction metadata in a local repository instead of direct updating master database. Since the metadata own intelligence in auto-configuration of the threshold of local stock volume and inventory sourcing, metadata also provide direction for the timing floor limit and filter of exception handling, any incoming event can be a trigger in hitting either the volume limit, timing limit or condition matched to initiate a batch job and taking all the accumulated records into a computation and use the aggregated summary to run a batch process.

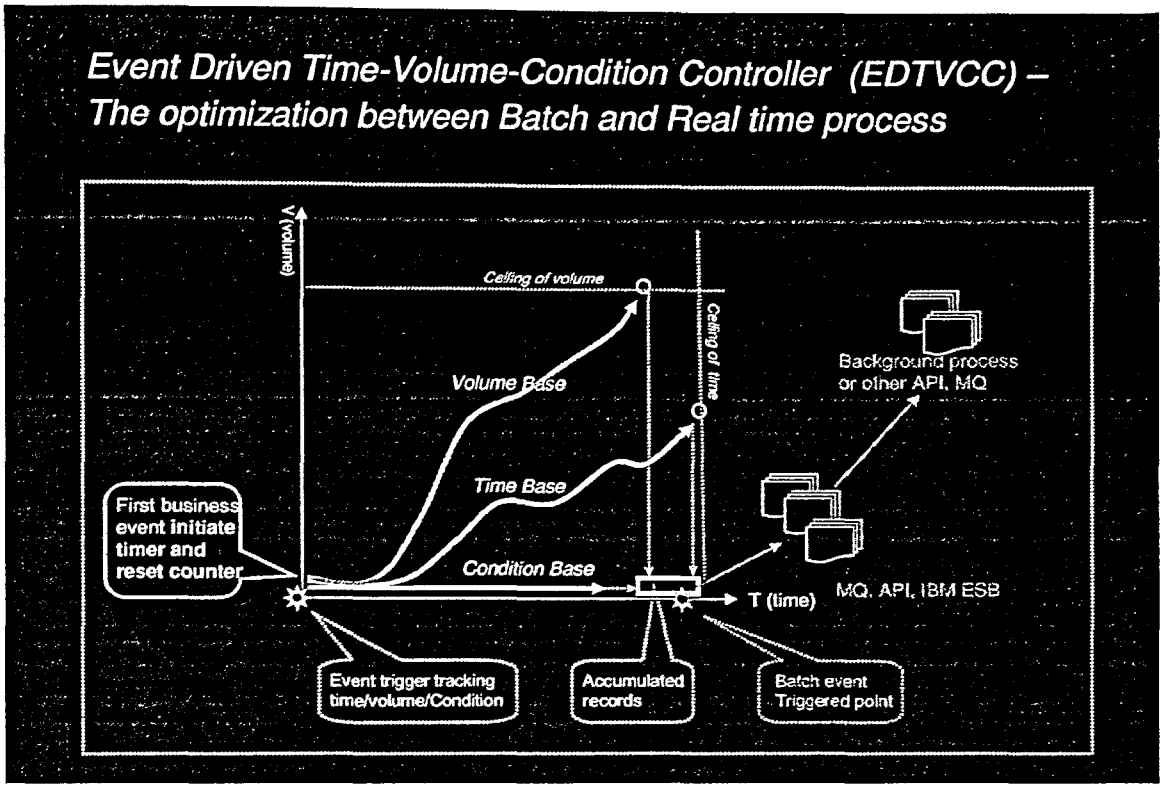
Motivation

The innovation of EDTVCC was tried to support Telecom industry in provisioning process, this is regarding the new customer, when CRM system change the contact into service account and billing account, system synchronize many sites in entering the account information, customer preference, configuration profile, provision status and many other system level provisioning management. Each of the incoming event can cause tremendous system resource consumption, timing delay, integrity issue, non-consistency and maintenance difficulty, among these the critical issue is the event only initiated in the first instance of creating service account, system will lose the further event trigger to take the first event forward across end-to-end process. Later implementation called Daemon process tried to solve the event issue by using "scheduled batch job" to check all the outstanding records of several tens, hundreds from millions records and submit to next Daemon for further process, this cause huge resource consumption, delay and errors due to process inconsistency and inefficiency. The motivation of EDTVCC was trying to solve these issues which were common in any industry of end-to-end process, when we take an insight of the business nature, either in SOA or non-SOA environment, event (the service initiator or succeeding trigger) has been poorly controlled and the big gap between process (either composite services in SOA or wired program in non-SOA) has been in lack of interoperability to establish an end-to-end process. The issue was related to workflow, however the common workflow package synchronize process unit by condition and status, however it cannot resolve timing, volume and performance issue as workflow has weak control of event initiated from real-time request, running into a batch process still need assessment of timing, volume and performance optimization through meta computation.

Now I believe it is clear that in a timing critical batch process, we need to take advantage of the initial event of "service consumer profile" (including all customer information, service option, preference and status) into a mediation of "service provider profile" for a metadata computing. The philosophy is once we grasp the meta-repository computing and register the "service" we can take top-down approach of chain-reaction which is

EDTVCC is the supplementary solution for SOA which may work on those interface or process which has sensitivity in control over event, time, volume and frequency particularly for those scenario that need comprehensive filtering, throughput control, inter-process external call and status tracking between source and destination, EDTVCC work as a mitigation solution.

Event Driven Time-Volume-Condition Controller (EDTVCC) –
The optimization between Batch and Real time process



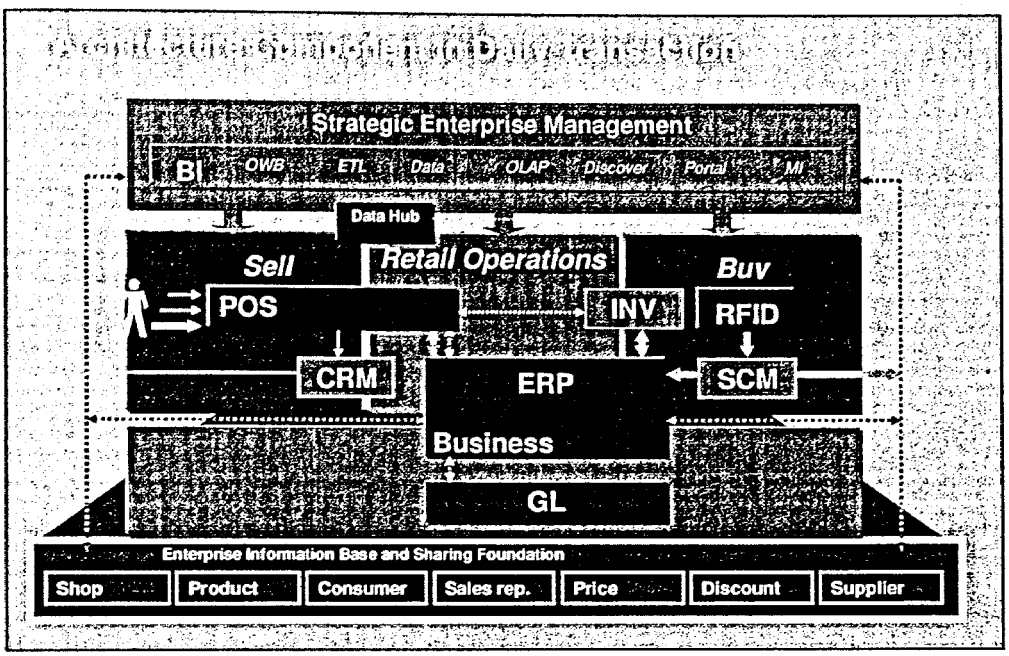
1. EDTVCC use configurable parameters to control the 3 dimensions optimization between - Time, Volume and Condition.

2. The incoming record accumulated in the queue until either time or volume reach their limitation (the configured parameter) or the condition meet the pre-set scenario, the batch trigger takes the whole batch to process the transformation, call the following background process, send the batch to remote destination and reset the trigger & counter.

3. EDTVCC is considered as a combination of both real-time and batch, flexible in parameter tuning and filtering. It can be a supplementary solution and being integrated into Enterprise Integration either as a solution or product base to increase data integration coverage.

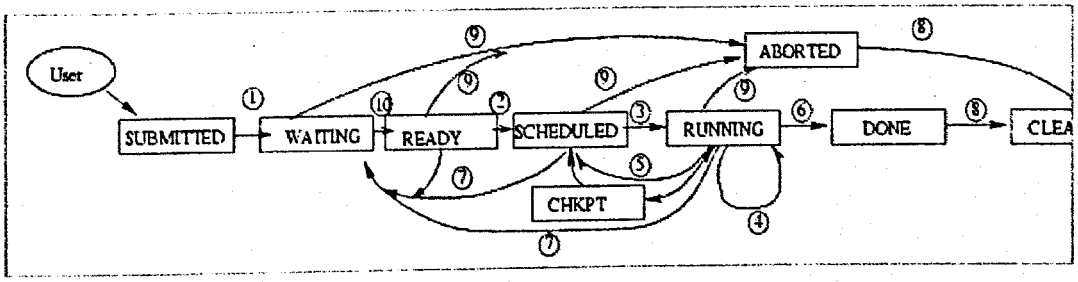
In a combination of real-time and batch process, event driven improve
the system operation. When the incoming event reach the ceiling (limitation)
of either volume (i.e. 100 records) or time (i.e. 5 minutes) the trigger will fire the
batch process to drive the data into succeeding process.
During the business peak hours, it takes short period to reach the volume limitation so the heavy tasks get released very fast, in the reverse case when the incoming event become slow in the midnight, the pending record guarantee being handled within a certain period, as both the volume and time parameters are configurable.

The EDTVCC operational environment, which is based on real-time batch process and when timing is not critical



Event Driven Stored Procedure (sample only)

```
CREATE PROCEDURE  .....
DECLARE
        CURSOR c1 IS SELECT field1 FROM Tbl1 WHERE field2 = 100;
BEGIN
FOR field1 IN c1 LOOP
        - - - - -
        IF acct_balance >= debit_amt THEN
        UPDATE accounts SET bal = bal - debit_amt WHERE account_id = acct;
        ELSE
        INSERT INTO Tbl2 VALUES  (filed1, field2, 'Insufficient funds');
        IF field1 IS NULL THEN RAISE Exc1;  -- raise exception
        END IF;
        Filed3 := (filed1 * 0.10) + (filed5 * 0.15);
        END IF;
                COMMIT;
END LOOP;
END;
EXCEPTION
WHEN Exc1 THEN ... -- process the exception
WHEN Exc2 THEN
INSERT INTO temp VALUES (NULL, NULL, 'Not found');
COMMIT;
WHEN ZERO_DIVIDE THEN
ROLLBACK;
WHEN VALUE_ERROR THEN
INSERT INTO errors VALUES ...
COMMIT;
WHEN OTHERS THEN
NULL;
```



1. A job submission request is passed by the UI to the RB.
2. A job submission request is passed by the JSS to the chosen CE.
3. The job is started on a CE.
4. The job has been affected by an event caused by the CE local resource management system, e.g. it has been suspended or its priority has been lowered.
5. The job is stopped and queued again in the CE.
6. The job has completed.
7. The job is passed again to the RB.
8. The bookkeeping information about the job is cleared.
9. The job is terminated.
10. A suitable CE has been found where the job can run.

SUBMITTED	The user has submitted the job to the User Interface.
WAITING	The Resource Broker has received the job.
READY	A Computing Element matching job requirements has been selected.
SCHEDULED	The Computing Element has received the job.
RUNNING	The job is running on a Computing Element.
CHKPT	The job has been suspended and checkpointed on a Computing Element.
DONE	The execution of the job has completed.
ABORTED	The job has been terminated.
CLEARED	The user has retrieved all output files successfully. Bookkeeping information is purged some time after the job enters this state.

T8

Database Partitioning and CLOB implementation

Abstract

This report provide the testing result and strategy for Oracle Partition implementation together with the Plan of replacing T29 CLOB column by parameter tables and T32 CLOB column by varchar2 data type. The aim of this implementation is for increasing the performance of Remedy (either ad-hoc query or Remedy process), enhance the archiving mechanism, saving storage and set Database's storage floor limit.

The design methodology is by taking advantages of Oracle Partition, to centralize those "wait_for_process" records to a small queue partition of around 2%, the automatical partition mechanism from the specified fields updating will "push" the records from the current state to queue partition, and pull them back once process completed and fields are updated to original. When the process handling 2% of data (instead of 100%) the performance and integrity is highly increased. The testing show that the smaller the range of queue partition, the better performance will be achieved. This is because the hit-rate increasement. The implementation need to combine both Partition solution with Parameter table (CLOB replacement) to achieve this result.

The other enhancement is Replacing CLOB by an external parameter table, this will improve the efficiency no matter how the CLOB is used in the application or not as when the application parsing with data caching the large object cause the buffer slowing down even they are not used in the computation. The results need be studied and strategy need be further investigated to transit to the Production, this will involved in;

- Application change, which is very small, but need to identify which Remedy user need to point T29 synomym to T29 view (queue partition) and which ad-hoc query user need to point their T29 synonym to T29_temp (full table), same for T32.
- Joinning of text table (T29_TEMP) and Parameter table need be revised.
- On-Line parsing of parameter for XML need be revised
- Partition need be applied to both text table and parameter table periodically (quaterly)
- T32 strategy is slightly different as T29 as external table may not required in diary portion

Presentation to EBT Telecom (Aug.25, 2003)

ORACLE

Strategy and Testing prepared by: Raymond Wu

Initial draft: August, 2003

Version: 1.2

Objective of Remedy Database Partition

1. **Enhance the Remedy Line Process Performance**
   This means we separate the data for Remedy process from the whole database to prevent full table parsing in Remedy process. Oracle partition provide the mechanism that when the partition "range" field (which is used in partition) is updated, the record will be transmitted to the partition of the new value (of range field), it's essential that we use this mechanism to "push" and "pop" those records (either insert or update/delete) into/from the queue stack (queue partition). Remedy only need to handle these records, so before the record is completed from Remedy, it will stay in queue partition, the performance will be increased due to "hit-rate" enhancement

2. **Ad-hoc query**
   As ad-hoc query is randomly distributed and there is no prediction for what sort of data range will be queried, this need investigation of the nature of the search conditions (but not the date-time), any fields concatenation which can be used in the data prediction of search criteria should be implemented for indexing purpose, together with enhancement of User Interface pre-sort grouping, should be formed a strategy of "high hit-rate" to achieve this aim.

3. **Archiving**
   One of the reason using Partition is for the Archiving purpose. Any field or concatenation of fields which can be used in Archiving should be used in sub-partition (or simply a query condition) to extract and archived to off-site for future usage or off-line query.

Data Flow in Partition

To minimize the Application modification, some database work has been done, including

1. Create view and synonym on the queue partition of T29 for Remedy or any other line process users
2. Create view and synonym on T29_temp (original T29 full table) for ad-hoc query
3. Implement XML parsing machine for one time conversion and on-line parsing

Fundamental Partition Testing

This test investigate the relationship between response time and Partition data retention
Partition retention means the percentage that queried records over the total record number
Table used in testing: T29 with 760,000 records

- Without partition (100% retention): Response time: 15 seconds
- Partition (50% retention): Response time: 12 seconds
- Partition (3% retention): Response time: 3.5 seconds

This fundamental testing show that when we condensed the data into 3% (by using partition) instead of original 100% (without partition), the performance improvement meet our requirements and query time will be cut down from 15 seconds to 3.5 seconds

Performance Evaluation

- Replacing CLOB column by Parameter columns
- Using Partition for Process Line Queue

T29 Testing Scenario

In the rough estimation, using Partition and Parameter table will be beneficial in the following advantages;

1. System Operation efficiency will be increased for 100 times in handling partition
2. Application process will be 10 times faster than current situation
3. Storage saving will be around 25% (from CLOB to Parameter table)

partition-schema-condition	1. CLOB	2. blank	3. blank + partition
a. Basic (no XML involved) 1.5% 10% (by-part)	63" 6' 39"	17" 1' 40"	8" 1' 1"
b. XML parsing 1.5% 10%	49" 82"	7.5" 25"	3.5" 15"
c. Remedy process	N/a	N/a	N/a
d. Device enhancement	N/a	N/a	N/a

Remedy Line Process and Storage management plan

Following 2 Remedy enhancements are expected to be completed by December 31.

- Enhancement-A: separation of line-process from Remedy storage
- Enhancement-B: Apply partition in Archiving

These enhancements are mainly trying to separate the Remedy process from the Storage, to increase the performance and lessen the line process complexity.

The other focus will be Storage management, to use Partition and Archiving to set the upper limit of DB storage, and use archiving to move the archived data to offsite, with special mechanism of switch control between active and offsite storage,

this enhancement will increase the Performance for those work-order related query and batch process, minimize the active DB storage and save the cost from disk space.

The enhancements will be divided into 4 stages with 4 checkpoints, each stage need special expertise and activities, those checkpoint need be committed before moving into next stage.

Check Point I (due August 1)

In this stage, enhancement-A need be completed for it's mechanism test

Enhancement-A

1. Complete Partition basic mechanism test
2. Ready for mechanism & performance test for the selected tables (T18, T21, H18, H21)
   - 2.1 DB – add 3 fields to the selected table (status, ins_date, upd_date)
   - 2.2 Remedy – Tables recreated by partition (ECL-Extract, reCreate & ins- Load)
   - 2.3 DB – Views/synonyms (use Synonym point to partition by same table name)
   - 2.4 UI – a general query mechanism for historical & active records

Enhancement-B

3. Study Remedy data integrity for archiving

Check Point II (due September 1)

In this stage,

Enhancement-A need be completed for QA and transited to Production

Enhancement-B need be completed for Function test of group-integrity archiving

Enhancement-B need be completed for Functional test of off-site search algorithm

Enhancement-A

1. Complete enhancement-A mechanism and Performance test
2. Produce Benchmark for enhancement-A (separate line-process from storage)
3. Rolling out enhancement-A to QA, and transit to production.

Enhancement-B

4. Readiness from Partition to Archiving
   Decide status code
   00 : waiting queue
   01-09 : in line_process
   10: complete (only this code is qualified for archiving)
   Decide field to synchronize group integrity: use "ins_date" initially
   Retain period before archiving, initially 6 months (tunable by multiple of 3)
   Study the query, and issue ping-pong (between active-offsite) mechanism
   Study extraction of full group integrity via ping-pong mechanism
5. Decide Group Integrity
   Scope and strategy of Group Integrity
   Prepare command and scripts to load data into partition (based on group integrity, the ins_date field) this is an one time job, need to load each full group into same partition
   Change application, that each new-entry inside a particular group will be time stamped accordingly
6. Off-site Archiving
   Test Partition rotate and exchange
   Test Transportable table space
   Test off-site search algorithm

Check Point III (due October 1)

In this stage,

Enhancement-B need be QA completed of group-integrity archiving

Enhancement-B need be completed for Function test of off-site search algorithm

Enhancement-B

1. Prepare synchronization between active server and off-site storage
2. Complete the function test of status rotate (00 to 10, 10 to 00 to 10)
3. Ready for the QA test for Group-Integrity search algorithm, so the user query result will be either 100% inside active storage or 100% inside of off-site storage
4. Complete function test of ping-pong mechanism (mechanism that if first search fail, system route the search to off-site storage)
5. Complete the function test of switch control of 'complete' case, so the completed case know

which zone (active or off-site archiving) the record need be sent to

6. Ready for the QA test of application that stamping the "ins_date" of new entry Record, ins_date will be based on the "root" of the associated group integrity

Check Point IV (due December 31)

In this stage,

Enhancement-B need be in Production, combined with Enhancement-A,

It produce a robustic mechanism of fas Remedy line_process, high performance of Work order quer and a lot of saving of disk Storage

Once Enhancement-A and B are in Production, a close monitoring, benchmark and auditing are required to keep a close watch of these new enhancements

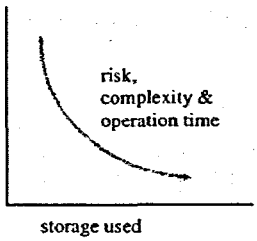
Oracle partition narrow down the computing target, when a potential client become customer account, the record will be processed inside the "active" layer (red color below) the record will be transmitted in different state associated with the "active" sub-partition, the transmission event trigged when a code/flag is changed and the performance and integrity will be improved as "physical" grouping is much more efficient that "logical" grouping.

The Purpose in ICT ETL process is trying to save storage by using Oracle Partition,

Initially it's expected to save 50% storage

Optimization is being reviewed, as the more storage saving the more risk and system complexity involved.

A maximum of 90% storage saving is possible, however the risk of operation, complexity and time spent in operation is rather high that may not in our full control



ETL process

- Extract: extract (from source – high end DB storage)
- Transformation: simplify data and compress
- Loading: load to target (low end partition DB)

Flow

A: data extract (pl/sql, dbms_output.put_line, shell-grep) by type, only info for settlement is extracted

B: 2 types flat files FTP to PC/DB and loaded (SQL*loader-direct) to partition (date/type)

C: PC/DB tape arch/purge

D: optional, flat file arch/purge

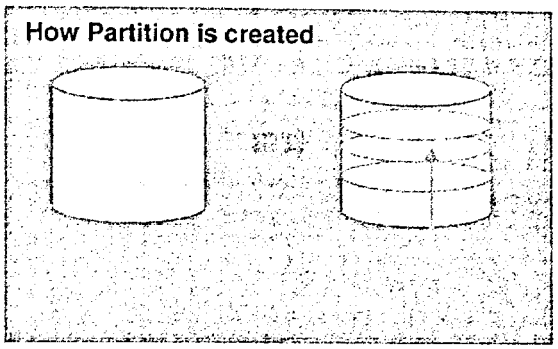
E: data which not extracted will stay for 7 months, data extracted will be arch/purged immediately

F: Unix/DB tape arch/purge

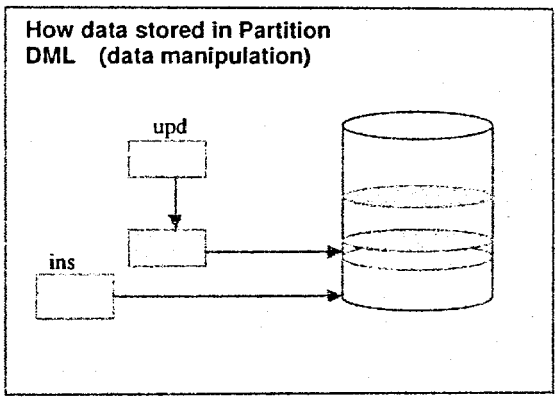
Terminology

- Archive: A, E
- Purge/Store: C, F
- Partition: B

4 major activities in Oracle Partition

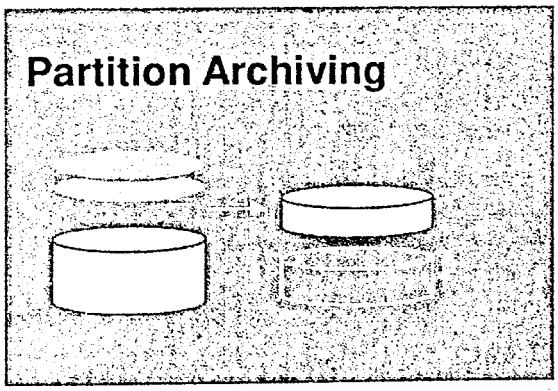


Partition 產生時機是於 Table create 時一併產生或 Table 產生後隨時因新的 Partition 需求,加上產生的
Partition can be created when table is created or adding new partition on an existing table create table test1 (key1 number, key2 varchar2(10), status varchar2(2), ins_date date, upd_date date) partition by range (status, ins_date) (partition q4_2002 values less than ('0',to_date('20030101','YYYYMMDD')), partition q1_2003 values less than ('0',to_date('20030401','YYYYMMDD'))) enable row movement;



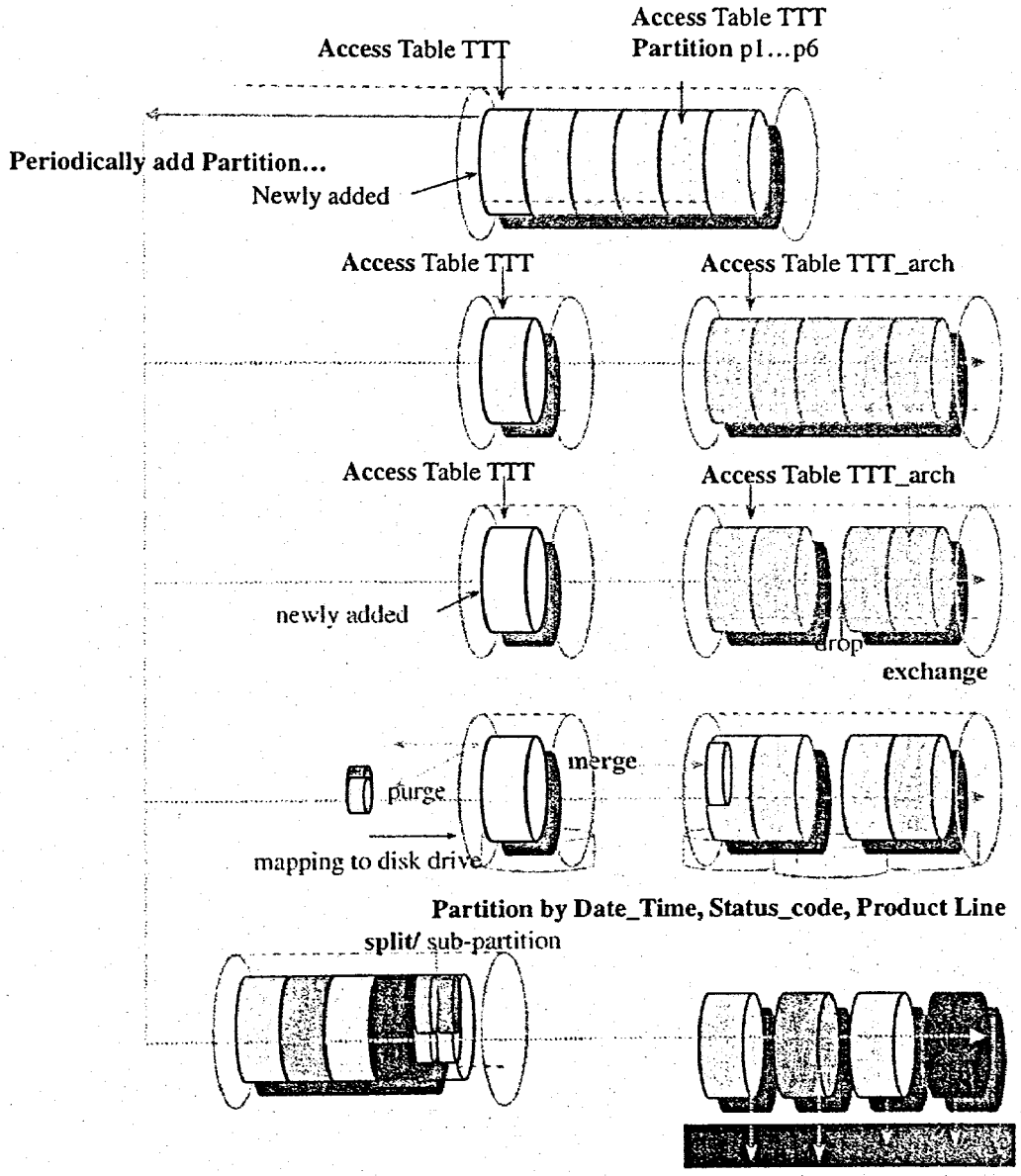
Data 與 Partition 之間的關聯取決於新 insert 的 Data 的 Partition filed(s)所著落的區間,既存 data 則在 partition field(s) 被 update 時 data 會因新的值所著落的區間而移動至新的 (partition field 值所對映的區間的) partition

The relationship between data and partition depend on the field of partition range - which range of the data belonging to. When a newly created record created (by insert), Oracle check the field(s) which used for Partition range, and decide which partition the data should go to
Data movement happen when the field(s) is updated movement between partition affected accordingly.



Partition query 除典型的 table partition query 外 以新增的 user 產生新的 view 及 synonym 將原先的 table name 對映至 view 所對映的 partition, 將於不需太多 AP 修改的前提下, 大幅提高系統的 performance
Typical SQL statement for partition query is;
select ... from <table> partition <partition> however, if application change need be minimized, to create new user, view and synonym (by using same object name) and map the newly created view to partition will enhance the performance with almost no application change required

Partition patterns in Application



Applications



A1 Integration strategy and case study in Financial industry

IEEE-AMS

A2 Case study of component Base Integration in the Financial Industry

IEEE Southeast

A3 Micro Process Design and Virtualization in service industry

ACM-EATIS

A4 Enterprise Integration in eGovernment industry

EG Inderscience

A5 Enterprise Component based interoperability in e-Government

Haamaha

A6 Component Design, Integration & Service Transf. in Financial Ind.

IIISCM

A7 Service Virtualization

IBM Corporation

A8 Financial System Architecture

Oracle Corporation

A9 Financial System Integration

Oracle Corporation

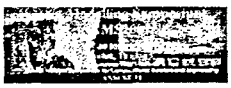
A10 Financial System Operation

Oracle Corporation

A1

A1: Integration strategy and case study in Financial industry

Publication: AMS (IEEE) Conference Proceeding, March 2007
Asia Modelling Symposium international conference ISBN 0-7695-2845-7 (P 384-387)
http://ieeexplore.ieee.org/search/searchresult.jsp?queryText=%28raymond
March 27-30, Phuket, Thailand
Publication Contact: Prof. David Al-Dabass
Conference Chair, AMS 2007
School of Computing & Informatics
Nottingham Trent University, UK david.al-dabass@ntu.ac.uk



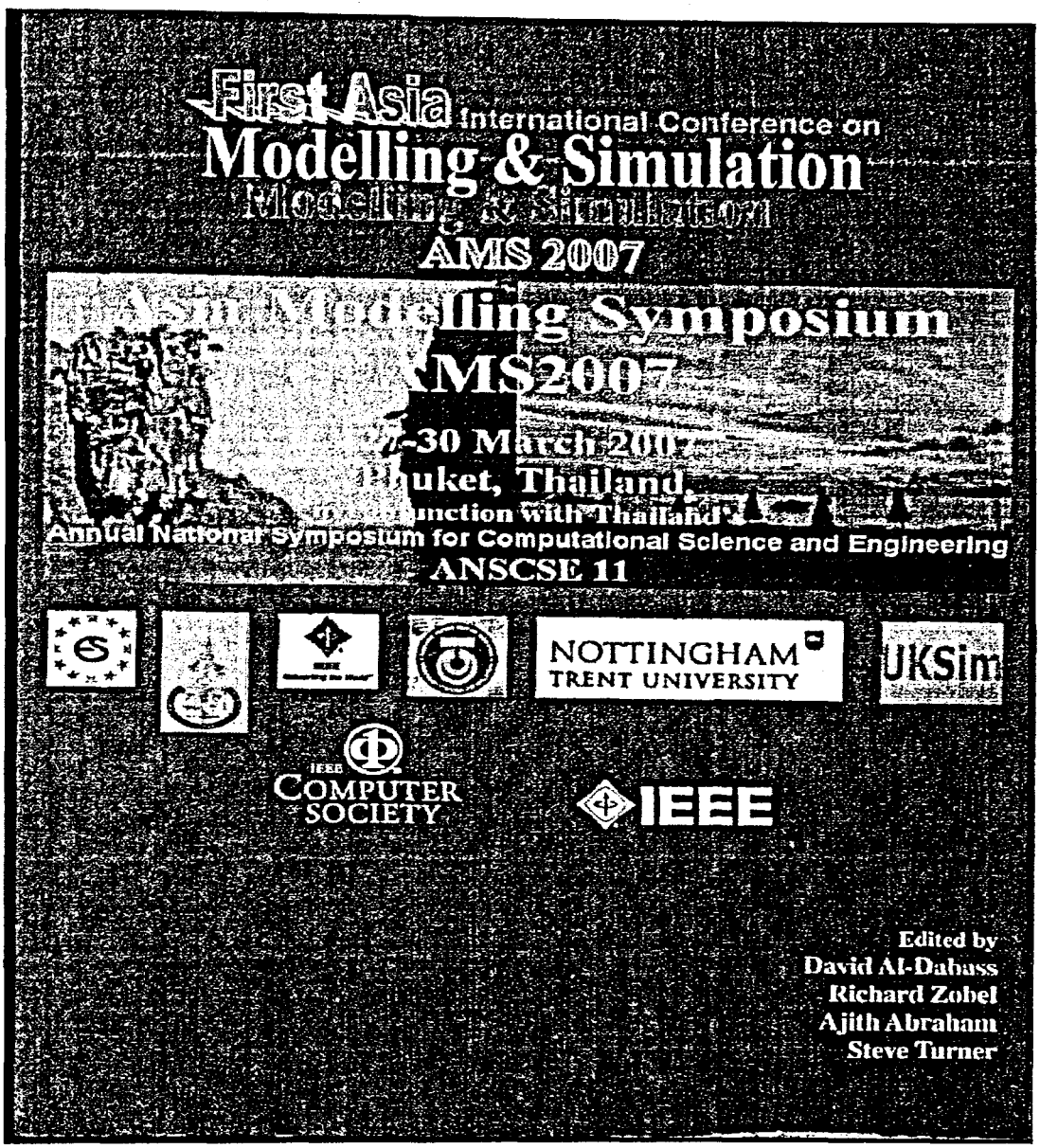
IEEE

From: david.al-dabass@ntu.ac.uk
To: "Raymond Wu" <rwu_academic@yahoo.com>
CC: david.al-dabass@ntu.ac.uk,
r.zobel@ntlworld.com, ASSJTurner@ntu.edu.sg, abraham.ajith@gmail.com

Dear Mr. Raymond Wu:

Congratulations - your paper #1569018281 for AMS2007 has been accepted.
The reviews are below or can be found at
http://edas.info/showPaper.php?m=1569018281, using your EDAS user name

===== Review =====
***** Relevance: Is the content appropriate for the conference themes? Yes (1)**
***** Technical Content: Is the technical content of sufficient quality? Medium (1)**
***** Originality: Does the technical content demonstrate sufficient originality for the conference? Medium (1)**
***** Writing Syle: Is the paper clear and easy to read and understand? Medium**



G. Intelligent Systems

Integration Strategy and Case Study in Financial Industry..... 384
Raymond Wu

Recursive Least Square and Fuzzy Modelling Using Genetic Algorithm for Process Control Application..... 388
Rizkan Zaiura Abdul Rahman, Rubayah Yusof, and Marzuki Khalid

Face Detection Using Artificial Neural Network Approach..... 394

INTEGRATION STRATEGY AND CASE STUDY IN FINANCIAL INDUSTRY

Raymond Wu
IBM Software Group
raymond@de.ibm.com

Abstract

According to the results of our recent research, enterprise integration in a component-based environment need validation, design and provision stages to achieve its business goal. Our current research has extended to the service virtualization which is the final stage in service provisioning. This research paper reviews our previous work and use examples in the financial industry to demonstrate how services are derived from business components, and how they are designed and produced in achieving Service virtualization.

Introduction

Financial industry is one of the high complexity industry among various sectors, and the severity is among those high risk areas. To eliminate the risk, our assumption is; the higher occupancy of common services within enterprise, the less suffering can be caused in each IT and business transformation, as technology and business innovation and transformation create opportunity while it may impact enterprise seriously if the transformation foundation is not solid, when they are enforced to transform their business or IT to be survival in next stage, weakness in enterprise skeleton and interoperability supported by common component may cause high risk in each transformation. To cope with these issues, we followed the enterprise integration strategy in our recent research which includes, validation of business and service component, service analysis and transformation, service design, information aggregation and service provisioning. [1]

Component Validation and Analysis in Enterprise integration

According to our previous research, enterprise integration is a metadata driven approach in the component-based environment. A management system composed of multiple distributed components implements a service such as content validation, transcoding, search or generation; and relies on metadata information to achieve success [2]. There are six steps in its component validation and analysis, validation of business components, service components, and physical components, common component parsing analysis, service aggregation parsing analysis, and determining implementation priority and skeleton. In component based validation process, three basic factors may influence the assessment of the usefulness of alternatives and the derivation of a satisfactory group solution. The first is the individual's role (weight) in the selection of a satisfactory solution. The second factor is the individual's preference for alternatives. The third factor is the criteria used for assessing these alternatives and making the decision [3].

Micro interoperability in enterprise integration and service virtualization

Common service and common information is a top-down approach which follows the nature of business implementation direction. From the technical perspective the idea was to eliminate the integration effort by using the "virtual organization" concept. From the organization perspective, this enhances the flexibility of service provisioning and the agility between organization and service; thus, the interoperability across organizations is increased.

Service Virtualization in Component based Financial Industry – Credit card transaction between card holder, merchant and interbanking process

We break down the business component into following use case scenarios for further development. The business scenario is related to credit card transaction between card holder, merchant and interbanking. We decompose the business components into following use cases, and use Figure 1. to illustrate the business scenario;

U-1 Buyer sends payment information to the merchant server. Merchant software must send and receive messages, encrypt and decrypt, store public and private keys, and request and receive certificates.

U-2 Merchant software takes payment information from the cardholder and sends it to the acquirer (the merchant's bank). The acquiring institution must receive and authenticate the payment information the merchant received from the cardholder.

U-3 Acquirer sends an authorization request to the issuer over the interbank network. Issuer sends the authorization response to acquirer.

U-4 Acquirer notifies merchant about the status of authorization; if the response is positive, merchant fills the order.

0-7695-2845-707 $25.00 © 2007 IEEE

U-5 Merchant presents the charge to the acquiring bank.

U-6 Acquirer sends a settlement request to the issuer.

U-7 Issuer charges the buyer's credit card account and at regular intervals notifies buyer of transactions and accumulated charges.

U-8 Buyer pays the charges to the bank. Acquirer credits the merchant's account.

In our current research, we are focusing the interoperability of service provisioning and virtualization. In the conventional Credit Card service, the interbanking doesn't maintain a "Fraud detection policy" service at account level, mostly this shortage came from conventional silo implementation and limitation of multi-sites interconnectivity so it was using a daily program to update the hard-coded matrix within the whole Authorization process.

The semantic interpretation from business to service, and from service to technical need decomposition process to work on those selected business components, so each business component can be decomposed into business use cases for further breakdown. Service components derived from business use case are the mediation between business and technical element which integrate all business semantic, interfaces and technical configuration into a logical component, initially all the service components are in atomic level [4] [5].

Service normalization means grouping all atomic service elements from fine-grained to coarse-grained. The conventional credit card service doesn't maintain a "fraud detection policy" service at the account level; it is maintained as a generic rule applied to all card holders within the whole authorization process. We further detailed the rational model in analyzing common component and aggregated services by using use case parser to determine enterprise component in forming the skeleton of integration. Aggregation is a kind of service normalization in grouping all atomic service element from fine-grained to coarse-grained component. We also introduced the component design and interconnectivity for how the component communicate each other and how the micro process of business semantic be derived into each component. Figure 2 further shows, the referential integrity of component interconnectivity for how the component related to each other.

Physical component design and aggregation

Figure 3 show the design of component and interconnectivity. The component element is the physical element that mediates service and information, the design of service components follows our previous strategy, by using business semantic injection to take the context into the semantic and micro processes, so that each component has its contract level responsibility in provisioning a subset of service, and the interface helps the coordination between components and metadata build the foundation of interoperability within the service layer and between layers.

The string $\Phi(r_n, r_m) = \Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m$ is a path from r_n to r_m, we call "service route" and $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$ a "composite service" between r_{i-1} and r_{i+1}, we use symbol $\Phi_{i, i+1}$ to denote composite service

When frequency $F(\Phi_{i, i+1}) = F(\Phi_i \times \Phi_{i+1})$ is higher than threshold σ, we aggregate the concatenated alphabet (part of the string) $\Phi_i \times \Phi_{i+1}$ into $\Phi_{i, i+1}$, this imply the original state r_i between r_{i-1} and r_{i+1} will no more exist for this composite service.

Component based information metadata interoperability

From information metadata perspectives, to achieve the service provisioning in single view of customer, we aggregate all fragmented data sources into metadata and business intelligence to build the service virtualization foundation based on the integrity between business, service and information. M. Schmalz argued that database schema and application code can be efficiently derived from various types of schema representations, particularly the relational model, and supports comparison of a wide variety of schema and code constructs [6].

Component based integration interoperability

The interoperability skeleton is an example for how the micro interoperability in service layer, aligned with component layer in micro mechanism, provisioning the macro service in business. The foundation of this skeleton is interconnectivity between service component by using common component and aggregated component. We further develop the service virtualization on top of this interoperability skeleton, which we call enterprise vertical integration, in extending the integration across industry and geographical areas, this is the virtualization service in horizontal direction, once industry of different regions build up their semantic interoperability in three metadata repositories (business, service and information), they can synchronize their business, operation and strategy, so that the limitation of business, geographical area, products, application, and data structure become transparent, this is what we call service virtualization.

Service virtualization

Once the vertical skeleton and service extension are built, the interoperability reflected from the business layer has the capability of integrating cross-organization and cross geographical constraints, as fully transparent services become possible if each system or organization follows an industry standard by using metadata strategy and an industry framework.

0-7695-2845-707 $25.00 © 2007 IEEE

There are three steps to reach service virtualization; [7]

1. Enterprise consolidation, to identify the unique source of truth within enterprise, and use its metadata within business, service and information as a corporative standard.
2. Vertical enterprise integration, as when other systems consolidated into the enterprise, there will be some metadata out of scope and need further integration effort to streamline the vertical integration, both of step 1 and 2 are shown in Figure 4. to demonstrate how we leverage horizontal resources into vertical integration
3. The final step is the service virtualization which is the objective of our enterprise integration, our metadata in business, service and information provisioning the services through industry and geographical locations in a transparent environment and in horizontal direction. This is shown in Figure 5.
4. Enterprise consolidation, to identify the unique source of truth within enterprise, and use its metadata within business, service and information as a corporative standard.
5. Vertical enterprise integration, as when other systems consolidated into the enterprise, there will be some metadata out of scope and need further integration effort to streamline the vertical integration, both of step 1 and 2 are shown in Figure 4. to demonstrate how we leverage horizontal resources into vertical integration
6. The final step is the service virtualization which is the objective of our enterprise integration, our metadata in business, service and information provisioning the services through industry and geographical locations in a transparent environment and in horizontal direction.

Conclusion

In enterprise service provisioning and virtualization, we constructed the skeleton of enterprise service with leverage to each layer and the interoperability of the layers. An example of service provisioning would be the case where several key industry vendors have built up an industry framework for service providers in that particular industry to follow, starting from business domain analysis and working through to component deployment. So each service provider benefits from the industry common services and common interface, particularly those B2B, B2C, merger and acquire cases. Any business activity becomes easier when the various parties involved have a common understanding and consensus on business terms, service definitions, interfaces and component specification, and these become transparent and common practices [8]. The important concept here is that once the metadata of business, service and information constructs an industry standard, the organization's vertical skeleton has the opportunity to integrate across multiple organizations for an easier global practice; this is a long term industry objective which we call service virtualization, and this is the pioneer work we are influencing Financial industry in achieving the objective of component based integration and service virtualization.

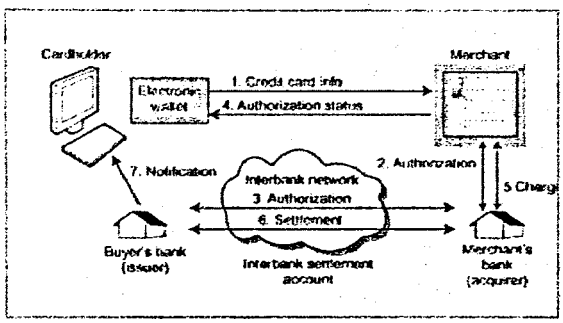


Figure 1. Business use case in Financial industry

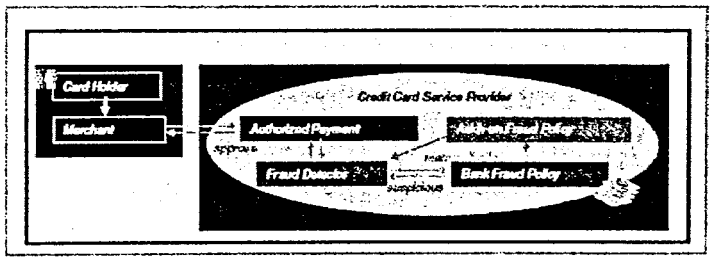


Figure 2. Loosely couple component adds-on solution for fraud detection

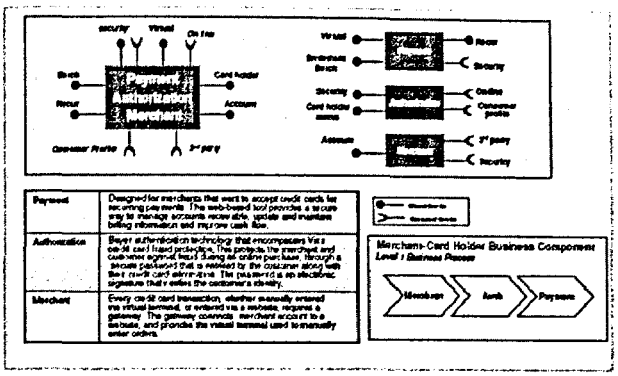


Figure 3. Component design in interconnectivity and composite services

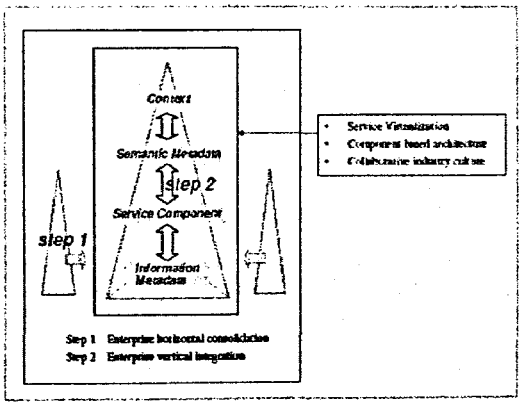


Figure 4. Enterprise consolidation and vertical integration

Reference

1 M. Lytras, (2006), "The Semantic Electronic Government: knowledge management for citizen relationship and new assessment scenarios", Electronic Government, an International Journal 2006 - Vol. 3, No.1 pp. 5 - 17

2. C. Foster, P. Stentiford, (1999), "Metadata - The Key to Content Management Services, Proceedings of the Meta-Data", The third IEEE Meta-Data conference

3. Jie Lu. Guangquan Zhang & Fengjie Wu, (2005), "Web-based Multi-Criteria Group Decision Support System with Linguistic Term Processing Function", IEEE Intelligent Informatics Bulletin June 2005 Vol.5 No.1

4. M. Staab, N. Stojanovic, R. Studer, and Y. Sure, (2001), "Semantic Portal - The SEAL approach. to appear in: Creating the Semantic Web.", MIT Press

5. O. Valkeapää and E. Hyvönen, (2006), "Semantic Annotation with Browser-based Annotation Tool", 1st Asian Semantic Web Conference

6. M. Schmalz, 2003, "EITH - A Unifying Representation for Database Schema and Application Code in Enterprise Knowledge Extraction" , 22nd International Conference on Conceptual Modeling

7. J. Hunter , (2001), "MetaNet - A Metadata Term Thesaurus to Enable Semantic Interoperability Between Metadata Domains", Journal of Digital Information, Volume 1 Issue 8

8. O. Zimmermann, P. Krogdahl, C. Gee, (2004), "Elements of Service-Oriented Analysis and Design", IBM Academy

A2

A2: Case study of component Base Integration in the Industry

Publication: Southeast07 (IEEE) Conference Procedeing
ISBN: 1-4244-1029-0

Publication Contact: Professor Gang Tao
Technical Conference Chair
Dept. of E.E., University of Virginia
gt9s@ece.Virginia.edu

IEEE SoutheastCon 2007

Date: Tue, 23 Jan 2007 13:18:11 -0800

From: gt9s@ece.Virginia.EDU Add to Address Book Add Mobile Alert

To: rwu_academic@yahoo.com

CC: gt9s@ece.Virginia.EDU

Subject: Your SoutheastCon 2007 Submission (Number 99)

Dear Mr. Raymond Wu:

On behalf of the SoutheastCon 2007 Technical Program Committee, I am pleased to inform you that the following submission has been accepted to the 2007 IEEE Southeast Conference:

Case study of component based integration in the industry

as a full paper.

The acceptance of your paper is based on the conditions that the paper will be properly revised according to the reviewers' comments, the final version of your paper will meet the posted manuscript guidelines, a full registration fee will be paid for the paper, and your paper will be presented at the conference.

Gang Tao
Technical Conference Co-Chair
IEEE SoutheastCon 2007

Reviewer's Scores

Originality: 4
Technical strength: 4
Presentation: 4
Overall evaluation: 4

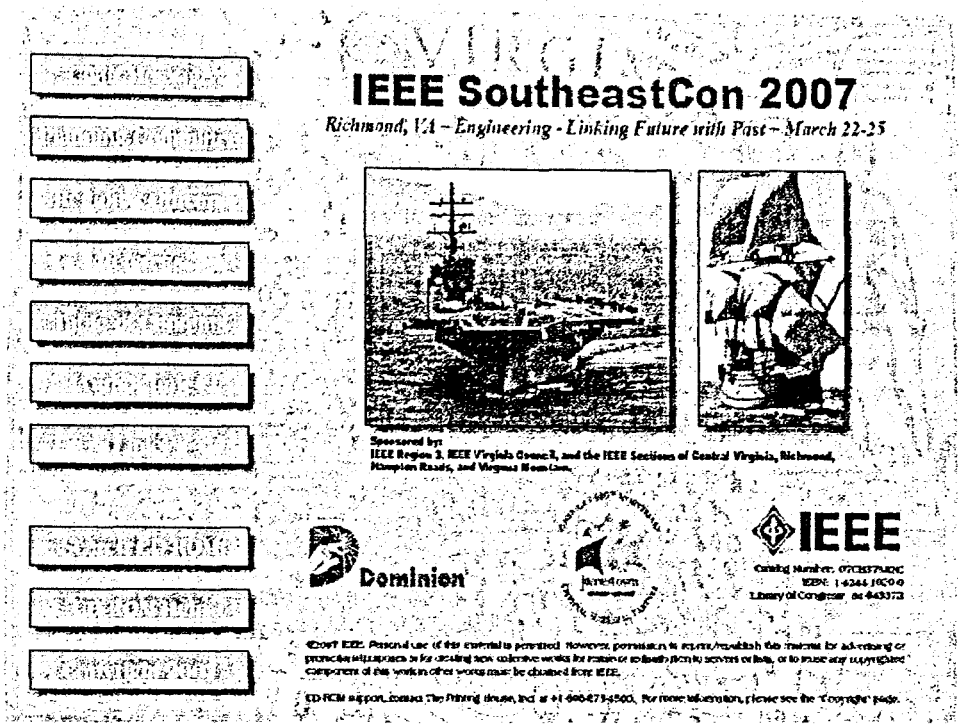


TABLE OF CONTENTS

Scroll to the title and select a Blue link to open a paper. After viewing the paper, use the bookmark "Go to Previous Document" to return to the same page in the Table of Contents.

Computing and Real-Time Systems

Case Study of Component base Integration in the Financial Industry 243
Raymond Wu

Case study of component base integration in the financial industry

Raymond Wu
IBM Software Group
raymond@de.ibm.com

Abstract

This research paper review our recent work in component based implementation and use example in financial industry to review the design process from micro component design to macro component based service virtualization. Financial engineering rely on inter-banking third party to mediate its services virtualization, there are two key processes play crucial roles in this mediation: the micro component design and macro metadata framework, the supplementary solutions of these two processes construct the enterprise integration interoperability and service virtualization which is the priority in financial industry.

Introduction

A typical financial project usually consists of domain implementation, architecture and shared service. The domain implementation group is divided into several teams to handle various LOB (line of business) as each team will have different areas of business domain knowledge; each team may in fact handle several business lines (i.e., use cases). It's very often the case that the individual LOB team doesn't have adequate concepts and communication to determine the common components, which include common services, common interface and standard data access and data models, we would like to review and list these common issues in enterprise integration with our proposals of resolution, this is shown in Figure 1.

i) Too many integration points cause implementation silos

ii) Data fragmentation with duplicate process without aggregation strategy [1]

iii) Common area is invisible and unable to reuse in a shard model

iv) Lack of priority and focus of strategy slow down business responsiveness

v) Weak enterprise interoperability cause high risk in industry transformation

In the recent requirements, the conventional credit card service doesn't maintain a robust "Fraud detection policy" at account level, so many transactions become invalid, the agility and responsiveness of making a proper change is still weak and many consumers suffer the serious issues that their transaction need be rolled back. For this reason, we launched add-on service component of Fraud Policy service to replace the conventional practice and dynamically retrieve card holder profile and the applied fraud policy to determine the authorization approval through inter-banking, so each account holder can have their security profile and preference for the card issuer or dealers to determine the individual fraud detection based on the transaction behavior and customer profile.

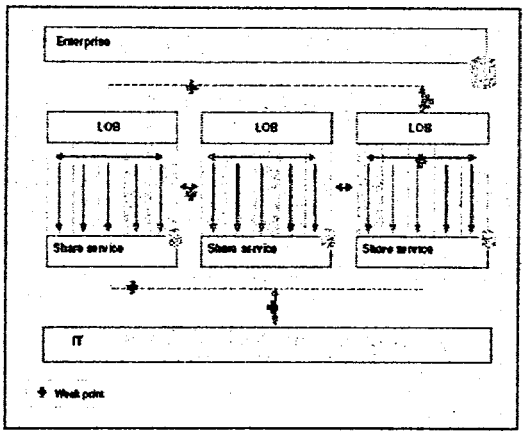


Figure 1. Project architecture common issue

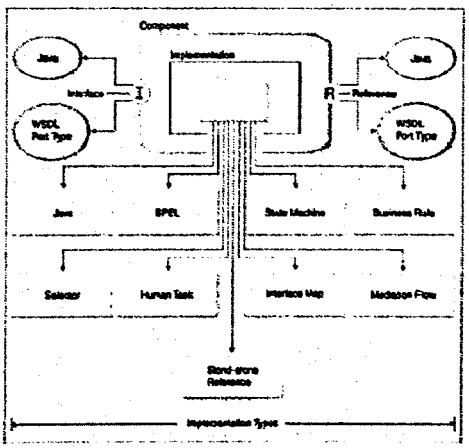


Figure 2. Service component architecture

1-4244-1029-0/07/$25.00 ©2007 IEEE

Literature review

Financial industry is one of the high complexity industry among various sectors, and the severity is among those high risk areas. To eliminate the risk, our assumption is; the higher occupancy of common services within enterprise, the less suffering can be caused in each IT and business transformation, as technology and business innovation and transformation create opportunity while it may impact enterprise seriously if the transformation foundation is not solid, when they are enforced to transform their business or IT to be survival in next stage, weakness in enterprise skeleton and interoperability supported by common component may cause high risk in each transformation. According to L. Yu et al., An obstacle to software reuse is the large number of major modifications that frequently have to be made as a consequence of dependencies within the reused software components. Component base implementation is the foundation in coping with this bottleneck [2] [3].

To cope with these issues, we followed the enterprise integration strategy in our recent research which includes, validation of business and service component, service analysis and transformation, service design, information aggregation and service provisioning. Within the enterprise integration life cycle, component validation and analysis, which we introduced in earlier research papers is the foundation in component design and industry service virtualization.

Component validation and analysis

Financial industry distinguished itself from other industry as time to market is critical, higher security requirements is mandatory, and inter-banking service need agility in mediating different parties into a common service infrastructure, for this reason, we apply SCA (service component architecture) shown in Figure 2, to deal with component design and interconnectivity in micro mechanism. Common service is a top-down approach follows the nature of business implementation direction.

Component validation and analysis consists of steps in validation of business components, service components, and physical components, common component parsing analysis, service aggregation parsing analysis, and implementation priority and skeleton. It will be followed by component design shown in Figure 3.

From the technical perspective the idea was to eliminate the integration effort by using the "virtual organization" concept, from the organization perspective, this enhances the flexibility of service provisioning and the agility between organization and service; thus, the interoperability across organizations is increased.

The realization of this top-down vertical integration approach requires robust mediation to support the inter-communication between component-component and component-repository, this is shown in Figure 4. In other words, the concept needs a macro framework and physical support in micro integration service bus such as the middleware in realization of run-time services.

Component design in financial example

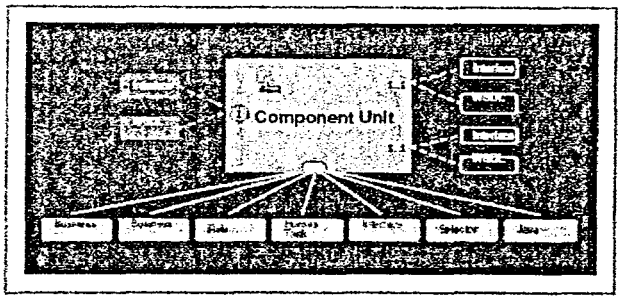


Figure 3. Service component design

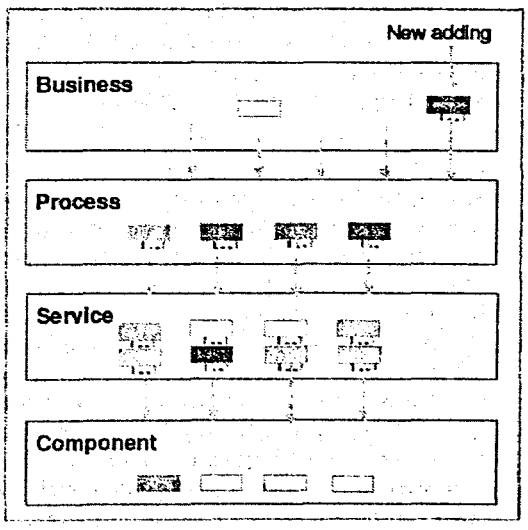


Figure 4. Top down component implementation

1-4244-1029-0/07/$25.00 ©2007 IEEE.

The micro process in component design follow the top-down direction and take the decomposition output to derive the service component. We use the inter-banking example of credit card industry to demonstrate the transaction between card holder (payer), merchant (payee), bank (transaction handler) and credit card authority (inter-banking). We decompose the business component of credit card transaction into several use cases: 1) Consumer contact merchant to encrypt and decrypt security keys, and request for certificates. 2) Payment is transferred from merchant to acquirer (the merchant's bank), and then to issuer through inter-bank network to authenticate the payment information. 3) Acquirer notifies merchant about the status for merchant to fill the order. 4) Merchant request acquiring bank regarding settlement. 5) Issuer charges the consumer's credit card and notifies card holder. 6) Consumer pay fee, acquirer then credits merchant for the exact fee.

The conventional design doesn't have the dynamic control to have customer insight while the issues were resolved in component based design. This is the service transformation and provisioning example which mediate the enterprise components in forming the skeleton. Service components derived from business use case are the mediation between business and technical element which integrate all business semantic, interfaces and technical configuration into a logical component, initially all the service components are in atomic level [4] [5]. In the component validation and analysis research, we applied group decision model in validation process, so business component with its associated service components are selected and validated through decision. We further detailed the rational model in analyzing common component and aggregated services by using use case parser to determine enterprise component in forming the skeleton of integration. Aggregation is a kind of service normalization in grouping all atomic service element from fine-grained to coarse-grained component.

We also introduced the component design and interconnectivity for how the component communicate each other and how the micro process of business semantic be derived into each component. The component element is the physical element that mediates service and information, the design of service components follows our previous strategy, by using business semantic injection to take the context into the semantic and micro processes, so that each component has its contract level responsibility in provisioning a subset of service, and the interface helps the coordination between components and metadata build the foundation of interoperability within the service layer and between layers.

- At the very beginning of a use case, the first event triggers service Φ_1

$$\varepsilon^1 : r_0 \times \Phi_1 \rightarrow r_1$$

- When the service event Φi ($i <> 1$) is read and triggers the service component, we express the state transformation as follows:

- $\varepsilon^i : r_{i-1} \times \Phi_i \rightarrow r_i$, we further derive the transformation as follows:

- $\varepsilon^{i+1} = (r_i) \times \Phi_{i+1} \rightarrow r_{i+1} = (r_{i-1} \times \Phi_i) \times \Phi_{i+1} \rightarrow r_{i+1}$

- Which is in equilibrium to $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$, and derives $\varepsilon^{nm} : r_n \times (\Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m) \rightarrow r_m$ $(n < m)$, we call this the "semantic event aggregator."

- As the final step of a use case, the final event trigger service Φ_n

- $\varepsilon^n : r_{n-1} \times \Phi_n \rightarrow r_n$, Φ_n is a terminator, an example of which is the settlement of a daily transaction.

- The string $\Phi(r_n, r_m) = \Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m$ is a path from r_n to r_m, we call "service route" and $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$ a "composite service" between r_{i-1} and r_{i+1}, we use symbol $\Phi_{i, i+1}$ to denote composite service

- When frequency $F(\Phi_{i, i+1}) = F(\Phi_i \times \Phi_{i+1})$ is higher than threshold σ, we aggregate the concatenated alphabet (part of the string) $\Phi_i \times \Phi_{i+1}$ into $\Phi_{i, i+1}$, this imply the original state r_i between r_{i-1} and r_{i+1} will no more exist for this composite service.

The foundation of this skeleton is interconnectivity between service component by using common component and aggregated component. We further develop the service virtualization on top of this interoperability skeleton, which we call enterprise vertical integration, in extending the integration across industry and geographical areas, this is the virtualization service in horizontal direction, once industry of different regions build up their semantic interoperability in three metadata repositories (business, service and information), they can synchronize their business, operation and strategy, so that the limitation of business, geographical area, products, application, and data structure

1-4244-1029-0/07/$25.00 ©2007 IEEE.

become transparent. this is what we call service virtualization. Once the vertical skeleton and service extension are built, the interoperability reflected from the business layer has the capability of integrating cross-organization and cross-geographical constraints, as fully transparent services become possible if each system or organization follows an industry standard by using metadata strategy and an industry framework. There are three steps to reach service virtualization: [6]

a). To identify the unique source of truth within enterprise, and use its metadata within business, service and information as a corporative standard, so when other systems and applications consolidated into enterprise repository, there is a standard and guideline to follow.

b). Vertical enterprise integration, as when other systems consolidated into the enterprise, there will be some metadata out of scope and need further integration effort to streamline the vertical integration.

c.) Service virtualization is the objective of our enterprise integration, we need to make sure our repository holding enough and accurate information within industry standard to integrate enterprise to external.

Conclusion

Financial industry is facing a big challenge within enterprise integration as the service concept is moving, global policy and regulation is changing and the high demands in transforming the system silos to an industry and global standard is not an easy task.

We gave the example of a typical issue in credit card scenario for how to narrow down the root issue within a small scope as a service component while maintaining its alignment to both business and technical components, this service component architecture speed up the change and enhance dramatically the integrity and responsiveness. An example of service provisioning would be the case where several key industry vendors have built up an industry framework for service providers in that particular industry to follow, starting from business domain analysis and working through to component deployment. So each service provider benefits from the industry common services and common interface, particularly those B2B, B2C, merger and acquire cases. Any business activity becomes easier when the various parties involved have a common understanding and consensus on business terms, service definitions, interfaces and component specification, and these become transparent and common practices [7].

Reference

1. Curtis, K., Foster, P. W., Stentiford, F. (1999), "Metadata - The Key to Content Management Services", Proceedings of the Meta-Data '99, The third IEEE Meta-Data conference, April 6-7, 1999, Bethesa, Maryland, USA
2. DSouza.(2001)." Model-Driven Architecture and Integration Opportunities and Challenges", Kinetium
3. L. Yu, (2005), "Categorization of Common Coupling in Kernel-Based Software," 43rd ACM Southeast Conference, 2005
4. M. Staab, N. Stojanovic, R. Studer, and Y. Sure, (2001), "Semantic Portal - The SEAL approach. to appear in: Creating the Semantic Web.", MIT Press
5. O. Valkeapää and E. Hyvönen, (2006), "Semantic Annotation with Browser-based Annotation Tool", 1st Asian Semantic Web Conference
6. J. Hunter , (2001), "MetaNet - A Metadata Term Thesaurus to Enable Semantic Interoperability Between Metadata Domains", Journal of Digital Information. Volume 1 Issue 8
7. O. Zimmermann, P. Krogdahl, C. Gee, (2004), "Elements of Service-Oriented Analysis and Design", IBM Academy

1-4244-1029-0/07/$25.00 ©2007 IEEE.

A3

A3: Micro Process Design and Virtualization in Service Industry

Publication: EATIS/ACM ACM-DL May 14–17th 2007, Faro, Portugal
Euro American Conference on Telematics and Information Systems,
ISBN: 978-1-59593-598-4

Publication Contact: Prof. Amine Berqia
Chair, EATIS 2007/6/11
Univ. Algarve, Portugal
chair@eatis.org

Date: Tue, 13 Mar 2007 20:26:02 -0600

To: rwu_academic@yahoo.com

Subject: [EATIS 2007] Your paper has been accepted!

From: "EATIS 2007" <chair@eatis.org> Add to Address Book Add Mobile A

On behalf of the EATIS 2007 - Euro American Conference on Telematics and Information Systems, I am pleased to inform you that your work tit

Micro Process Design and Services Virtualization in Enterprise Integration

has been accepted. Congratulations!

Yours sincerely,

Dr. Amine Berqia
EATIS 2007 Program Committee Chair

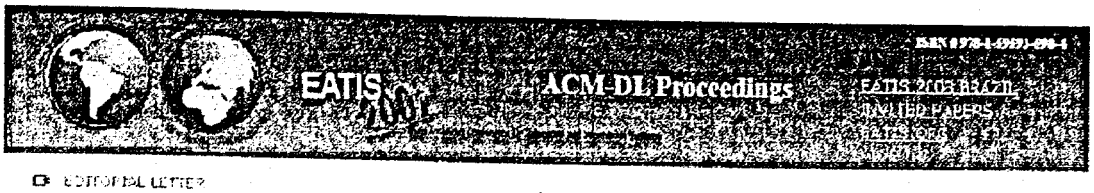
****

It seems to be a good paper, but some practical application is missed. In my opinion this paper is too theoretical.

****

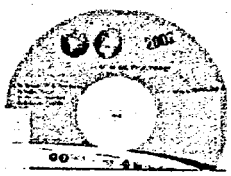
The theory sounds quite promising, however I would have liked to have found more concrete examples to help explain why each part of your process is necessary, and also proof-of-concept comparison with simila competing processes. Formulae are not enough for me.

As is, the paper would be a fine contribution to a tutorial session.



EDITORIAL LETTER
KEYNOTE SPEAKERS
TUTORIALS, WORKSHOPS AND SEMINARS
EATIS 2007, FULL PAPERS
EATIS 2007, SHORT PAPERS
ORGANIZATION
CONFERENCE SCALE
PAPER TOPICS





FULL PAPERS

ID 30 - Micro Process Design and Virtualization in service industry
Raymond Wu, IBM
Germany

Micro Process Design and Virtualization in the Service Industry

Raymond Wu
IBM Software Group
raymond@de.ibm.com

Abstract

In this research we deal with the design stage by introducing the micro process and interconnectivity concept as the foundation of the service component design. Service virtualization is a new area requiring our attention as in the service industry all enterprise integration and industry silos need to be consolidated and form a main route. The service grid concept will provision a transparent and highly integrated framework to provide global service infrastructure which eliminates constraints in geography and the system we use. In this research paper we introduce a conceptual service provisioning with extension to interconnectivity as the service foundation. We further explore service virtualization based on the concept we introduced and on micro process design foundation.

Key Words: Enterpise, integration, component, metadata

Introduction

Our research in enterprise integration has been divided into four areas: decision, analysis, design and virtualization, and the scope of this research paper is design and virtualization. The micro process of enterprise integration differentiates itself from the macro process in that the macro process is working on the configuration of components at an architectural level while the micro process deals with components both internally and in their interference in component fabrication, with how they are integrated in micro mechanisms and with how they work together in provisioning the coherence of a business goal. This includes the integration of common components and service aggregation in micro processes. Those components with a high frequency of utilization are termed common components. Both aggregated and common components need to be handled carefully; the steps for achieving this level are shown in Figure 1.

A component-based model builds the foundation for both business and service components, storing the intelligence of mapping and interconnectivity information in a repository in metadata form and strengthening the foundation of vertical enterprise integration. There are three areas of micro process in enterprise integration:

Area 1. Micro mechanism of service component fabrication and interference

Area 2. Micro process in service mediation and component transformation within the service layer

Area 3. Micro and macro interoperability in enterprise integration and service virtualization

The component element is the physical element that mediates service and information. The service component consists of interface protocol, business rules, semantic intelligence and referential information. It is important to note that (a) components are encapsulated for reuse; (b) components may be wired together as composite services; and (c) business objects are the data flowing on wires between components.

As the objective is to build rule-based enterprise aggregated services, we introduce the "state" concept to represent the intermediate "enterprise status" between transformation and post-transformation. The state concept in our micro process transformation is not merely dealing with a component or a service, but with the service layer and its coherence of enterprise interoperability of business and information. We extend this concept and apply event-triggered service execution to leverage the reading behavior of the automata turing machine while using service aggregation of composite services to represent the language acceptable by the machine.

The importance of determining the common components and service aggregation is in building up an enterprise skeleton of vertical interoperability between business, service and information. This task is crucial, as in the business domain analysis stage, the impact area and scope of business components are minimized in decision making. Additionally, the business component is deconstructed into service components and physical elements for top-down alignment while the common component is not simply a symbolic service but instead requires a bottom-up approach to work on common areas; this serves to fine tune the common components and interfaces from the perspective of enterprise integrity and performance. The service aggregation is also important as the top-down deconstruction followed by bottom-up composition forms

the skeleton of vertical interoperability and potential component-based implementation of the service process.

Literature Review

According to P. Kruchten, a component is a non-trivial, nearly independent and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture. A component conforms to and provides the physical realization of a set of interfaces. First, a component is non-trivial; it is functionally and conceptually larger than a single class or a single line of code. Typically, a component encompasses the structure and behavior of a collaboration of classes. Second, a component is nearly independent of other components. It rarely stands alone. A given component collaborates with other components and in so doing assumes a specific architectural context. This architectural context is driven in large part by the implementation we choose. Third, a component is a replaceable part of a system. A component is substitutable for any other component that realizes the same interfaces. This characteristic helps during development in that parts of a system can be stubbed, sketched and then replaced by mature, robust implementations. It also supports the evolution of a system even after it has been deployed, by making it possible to upgrade and evolve parts of the system independently. Fourth, a component fulfills a clear function.

In recent years, numerous projects have been undertaken by the many players and stakeholders in the information community to achieve interoperability among different metadata schemas and their applications. Ideally, a uniform standard approach would ensure maximum interoperability among resource collections. If all participants of a consortium or repository were required to use the same schema, such as the MARC (Machine-Readable Cataloging) format or the Dublin Core (DC), a high level of consistency would be maintained. This, of course, has been the approach in the library community for over a century and is the ultimate solution to the interoperability problem. However, although it is a conceptually simple solution, it is not always feasible or practical, particularly in heterogeneous environments serving different user communities where components or participating collections contain different types of resources already described by a variety of specialized schemas. The uniform standardization method is only viable at the beginning or early stages of building a digital library or repository, before different schemas are adopted by the participants. Examples include the MARC standards used in union catalogs of library collections and the Dublin Core-based Electronic Theses and Dissertations Metadata Standard (ETD-MS) used by members of the Networked Digital Library of Theses and Dissertations (NDLTD).

Area 1: Micro process component design

The component element is the physical element that mediates service and information. The service component consists of interface protocol, business rules, semantic intelligence and referential information. It is important to note that (a) components are encapsulated for reuse; (b) components may be wired together as composite services; and (c) business objects are the data flowing on wires between components.

The design of service components follows our previous strategy by using business semantic injection to take the context into the semantic and micro processes; by this means each component has its contract-level responsibility in provisioning a subset of services. This interface helps the coordination between components and metadata to build the foundation of interoperability within the service layer and between layers. In the following paragraph we take the results from component design into the service level while applying automata theory to build up the rule-based service aggregation at the enterprise level. As we are still at a pioneer stage in applying automata theory in service provisioning and aggregation, we understand this cannot be accomplished within a short period. Further research efforts will be undertaken following the current work. [1]. SCA (service component architecture) is the framework for micro process design of service components; SCA deals with component design and interconnectivity in micro mechanism. The service interface specifies SP (service provider) implementation and is invoked by the SC (service coordinator). The SC has the capability to invoke SPs through the service bus without needing to know what SPs are and how the interface was configured.

Before the design of the service component and transformation, we take direction from the business component by using a top-down approach to derive the component design while at the same time streamlining the alignment between service components and physical components by using both top-down and bottom-up approaches. Most of the bottom-up approaches are a componentization process working from existing information components, business objects, codes, system metadata or database access protocols. Once we construct the alignment between business, service and information and have the service component design and interconnectivity completed, we actively work on service layer transformation. The reason this step is crucial is that the goal of vertical enterprise integration is to create interoperability between layers from both micro and macro perspectives, with both business and information layers working as a foundation to support the formation of the service layer which provides the actual services. Once the

service component has completed its design with alignment to business and information, the focus can be shifted to micro interference within the macro view. In micro process ontology, we introduced SCA, which deals with component design and interconnectivity in micro mechanism and consists of three main areas. The service component is produced in the following three criteria:

1. Interface

The service interface specifies the SP implementation and is invoked by the SC. The SC has the capability to invoke SPs through the service bus without needing to know what SPs are and how the interface was configured.

Luca et al. demonstrated their surveying results from the theory behind formalizing rich interfaces for software and hardware components. Rich interfaces specify the protocol aspects of component interaction. Their formalization, called interface automata, permits the compiler to check the compatibility of component interaction protocols. Interface automata support incremental design and independent implementability. Incremental design means that the compatibility checking of interfaces can proceed for partial system descriptions without knowledge of the interfaces of all components. Independent implementability means that compatible interfaces can be refined separately while still maintaining compatibility. We used both of these methodologies in component interface design [2].

2. Implementation

The service provider implementations in an SCA are called service components and include:

- Java objects

A Java object is an implementation of a component in Java which has either a WSDL (web service definition language) or Java interface type. A Java object normally works in a mediation module.

- Business processes

A business process is a combination service made up of service components with a specific order to achieve a business goal; a business process should not be equal to a use case, rather it can be a result of deconstruction and can be composed of use case.

- Variables

Variables store the data used by a business process. The messages represent the writing, and the paper represents the variable.

- Microflow or short running process

A microflow is a process that is contained within a single transaction. This is ideal for situations where the user is expecting an immediate response.

- Long run processes

A long running process executes over an extended period of time and is much more flexible and resilient than a microflow. Interruptible business processes and asynchronous business processes are examples of long running processes.

- Human tasks

A human task is a unit of work done by a human which involves the interaction with other services; it thus becomes a task within a larger business goal.

- Business state machines

A state machine is an event-driven business transaction in which external operations trigger changes that guide the transaction from one discrete mode to another. Each mode is an individual state, and this mode determines what activities and operations can occur.

- States

A state is one of several discrete individual stages that are organized in sequence to compose a business transaction.

- Composite state

A composite state is an aggregate of one or more states. It is used to deconstruct a complex state machine map into an easily comprehended hierarchy of state machines.

- Business rules

A business rule is anything that captures and implements business policies and practices. A rule can enforce business policy, make a decision, or infer new data from existing data.

3. Metadata

The parameters transmitted in and out from a service are modeled as service data objects implemented by Java class. SDO (service data object) complements SCA by providing a common way to access many different kinds of data. The specification reduces the skill levels and the time required to access and manipulate business data. Today, a great many APIs are used to manipulate data. These APIs tend to tightly couple the source and target of the data; this makes their usage error-prone and subject to breaking as business requirements evolve. However, SDO makes APIs easier to use and thus realize their value without having to code directly to them.

The micro process starts with the mechanism for how the component is designed; this involves taking the business semantic of the macro level with coherence of enterprise interoperability between each layer. The service component possesses the capability of semantic handling, interfacing and metadata integration as shown in Figure 2.

Area 2: Service transformation

D. Roman et al. in their "Web Service Modeling Ontology (WSMO)" state that semantic services need to meet the following requirements: web compliance, ontological base, strict decoupling, centrality of mediation, ontological role separation and execution semantics [3]. These requirements were addressed in our research and the strategy is identical to that which we proposed.

We describe service transformation as the staging and intermediate area for taking micro design into the architecture level. Service transformation accommodates component design in provisioning the service, as component design takes macro vision to implement the component. Once this step is completed, our focus can shift

to the transformation of how they interact with each other in achieving a macro objective. There are two steps in service transformation: The first step is called mediation, which is the "interconnectivity reason" for the interaction of components, and the second step is called interoperability, the "skeleton strategy" in which the mediation is taken to the enterprise level to construct the enterprise's common services.

Mediations can be explicitly configured within a solution. As well as being attributes of the services, policies can be set by the solution administrator for a single interaction or set of interactions [4]. Once the vertical skeleton and service extension are built, the interoperability reflected from the business layer has the capability of integrating cross-organization and cross-geographical constraints, as fully transparent services become possible if each system or organization follows an industry standard by using metadata strategy and an industry framework. An example of this would be a case in which several key industry vendors have built up an industry framework for service providers in that particular industry to follow, starting from business domain analysis and working through to component deployment. Each service provider would thus benefit from the industry common services and the common interface, particularly those involving merger and acquisition cases (B2B, B2C). Any business activity becomes easier when the various parties involved have a common understanding and a consensus on business terms, service definitions, interfaces and component specification, particularly if these become transparent and common practices.

Mediation patterns manipulate messages in-flight on the bus by requests or events. According to IBM service mediation, there are several basic patterns for mediations; the following more complex patterns can be built by combining simple patterns:

- Protocol switch: Enables service requesters to dispatch their messages using a variety of interaction protocols or APIs such as SOAP/HTTP, JMS and the MQ Integrator (MQI).
- Transform: Translates the message payload (content) from the requester's schema to the provider's schema. May include enveloping, de-enveloping or encryption.
- Enrich: Augments the message payload by adding information from external data sources such as customization parameters defined by the mediation, or from database queries.
- Route: Changes the route of a message, selecting among service providers that support the requester's intent. Selection criteria can include message content and context as well as a target's capabilities.
- Distribute: Distributes the message to a set of interested parties and is usually driven by subscriber interest profiles.
- Monitor: Observes messages as they pass through the mediation unchanged. Can be used to monitor service levels, assist in problem determination or meter usage for subsequent billing to users.
- Correlate: Derives complex events from message or event streams. Includes rules for pattern identification and rules that react to pattern discovery; for example, by generating a complex event derived from the content of the triggering event stream.

The interoperability skeleton shown in Figure 3 is an example of how the micro interoperability in the service layer, aligned with the component layer in micro mechanism, provisions the macro service in business. The foundation of this skeleton is achieving interconnectivity between service components by using a common component and an aggregated component. We further develop the service virtualization on top of this interoperability skeleton, which we call enterprise vertical integration, in extending the integration across industry and geographical areas. This is the virtualization service in a horizontal direction; once the industries of different regions build up their semantic interoperability in three metadata repositories (business, service and information), they can synchronize their business, operation and strategy even if they have their local implementation extended from the interoperability skeleton. This demonstrates the resolution so that the limitation of business, geographical area, products, application and data structure become transparent. This is what we call service virtualization; we predict that virtualization of services will become a priority in industry as semantic interoperability is in high demand to solve the issues in both vertical interoperability and horizontal interconnectivity.

In our previous research on metadata interoperability we revealed, in relation to both political and rational models, how the components are derived through the validation process and how the strategy of component implementation achieves layer interoperability. In this paper we extend our research in component design and technical implementation of both internal and interface component factors [5]. Common services and common information are all interoperability strategy, which is a top-down approach by following the nature of business implementation direction. From the technical perspective the idea was to eliminate the integration effort by using the "virtual organization" concept. From the organization perspective this enhances the flexibility of service provisioning and the agility between organization and service, thus increasing the interoperability across organizations. The realization of this top-down vertical integration approach requires robust middleware to support the inter-communication between component-and-component and component-and-repository. In other words, the concept needs physical support from an integration service bus such as the middleware in run-time services.

Area 3: Service virtualization

Service virtualization is our objective in enterprise integration, although interoperability is the required intermediate step in achieving this goal. Service virtualization means the internal enterprise has a framework and strategy to – that the integration is not limited to layer-to-layer or across layers, instead it follows the nature of business and implements all semantic metadata in a component-based foundation for both micro and macro designs.

From the external enterprise perspective, service virtualization means industry follows a common framework with common services and components for service consumers to share the same protocol, interface, service components and information in a transparent environment.

Enterprise interoperability is a crucial checkpoint for service virtualization, as in the real world industry has many different methodologies, systems and terminologies, and a variety of styles of architecture and integration. Our approach has been set as the following three steps:

1) Ontology: The first step is an ontology to take all methodology, systems, terminology, architecture styles and integration into the main route, with insight into their interference and consolidation.

2) Framework: A framework, life cycle roadmap and strategy in achieving enterprise interoperability is a critical milestone. The reason is that when we push an implementation silo into the main route, we need a multi-dimensional mechanism to handle both micro and macro interoperability to integrate the assets within the ontology.

3) Provision: The final step is integration and tuning between theoretical research and common industry practices; minimization of the gap between our strategy and industry is the realization step in achieving service virtualization.

The interoperability approach of these three steps is shown in Figure 4.

In our previous terminology, those components with a high frequency of utilization are termed common components. At an enterprise level exceptions in handling a particular requirement are normally not allowed in order to maintain high integrity within the enterprise; therefore, consensus and enterprise standards are usually required at the outset prior to the design. As aggregated services and common components are critical in forming the enterprise skeleton, the design and aggregation process must be cautious from an interconnectivity perspective. This implies that high integrity, optimization, grain size, dimension in service provisioning, visibility, traceability, performance and other factors are areas which must be brought into consideration. Once all of these skeleton elements are well specified and designed, the "main route" needs to be outlined at a state-of-the-art level. What we term the main route is based on the top of the enterprise skeleton we build; the value-added service may mediate this aggregation and form a long-running process as a skeleton for a business line. One example is in the credit card business in which aggregated services have been built between the card holder and card issuer; in this case the same aggregation is also being formed between the merchant and the bank. The mediation service of an interbanking fraud detector component has been designed by reading fraud profiles in real-time transaction environments to mediate the individual services into a business main route; this highly enhances the service provisioning in a particular dimension.

The parameters transmitted in and out from a service are modeled as service data objects implemented by Java class. SDO complements SCA by providing a common way to access many different kinds of data. The specification reduces the skill levels and the time required to access and manipulate business data. Today, a great many APIs are used to manipulate data. These APIs tend to tightly couple the source and target of the data; this makes their usage error-prone and subject to breaking as business requirements evolve. However, SDO makes them easier to use and thus for their value to be realized, without having to code directly to them.

Aggregated services and common components should not be tightly coupled; instead, their dependency and interconnectivity should be based at a parameter and metadata level and their design and interfacing specifications should be fully visible for future development. The reason for this is that the more significant role these skeleton elements play, the higher the complexity of interconnectivity, business criticality and access frequency, which can result in significant impact if a defect is detected too late. In many instances, a defect is a kind of binding between components instead of a coordinate from aggregation. A service aggregation needs to be implemented in an empirical rule-base in the same way that a phrase needs words, grammar, semantics and a certain level of common recognition. It should not be allowed to be handled for individual needs differently or randomly; this is the reason it should be designed and maintained at the enterprise level as a single view-of-enterprise solution.

The micro process starts with the mechanism for how the component is designed; this involves taking the business semantic of the macro level with coherence of enterprise interoperability between each layer. Before the design of the service component, we take direction from the business component by using a top-down approach to derive the component design while at the same time streamlining the alignment between service components and physical components by using both top-down and bottom-up approaches. Most of the bottom-up approaches are a componentization process working from existing

information components, business objects, codes, system metadata or database access protocols. Once we construct the alignment between business, service and information and have the service component design and interconnectivity completed, we actively work on service layer transformation. The reason this step is crucial is that the goal of vertical enterprise integration is to create interoperability between layers from both micro and macro perspectives, with both business and information layers working as a foundation to support the formation of the service layer which provides the actual services. Once the service component has completed its design with alignment to business and information, the focus can be shifted to micro interference within the macro view. In our previous research on metadata interoperability we revealed, in relation to both political and rational models, how the components are derived through the validation process and how the strategy of component implementation achieves layer interoperability. In this paper we extend our research in component design and technical implementation for both internal and interface component factors.

Common interface is also a means to achieve common components. S. Melnik proposes a universal interface which avoids common models and languages; this is the Generic Interoperability Framework which was developed to facilitate the integration of heterogeneous information systems. Within this framework individual communication protocols, data manipulation languages and data are represented in a generic manner while preserving their ontological variety; this process indicates how a mediation infrastructure can be built using canonical wrappers. operational requirements of integration and layered architecture for component design [6].

The iteration through each use case will be tracked for its path and service statistics so that once we parse all use cases we should be able to see the frequency of use of each service, as well as the path and dependency between services. Where there is a high frequency of services we use the term "common services" and focus on the potential for reuse and minimization of the integration interface. In the case of service paths in which several services have a common sequence and appear in use cases a certain number of times past the threshold, the services can be aggregated to a composite service or potential process. This needs to be designed at an enterprise level so that each organization has the same concept of and specification for this common service. Service aggregation means a composite component consisting of at least two components that form a unique sequence of execution order with a common interface within their boundaries; this composite component will be treated as a coarse-grained service in provisioning the upward aggregation of services.

Conclusion

In component design we introduced the SCA (service component architecture) to further clarify the micro mechanisms within components as well as component interconnectivity. We classified component design into interface, implementation and metadata areas to guide the design from the internal to the external. We proposed that micro process component design needs to be the foundation of service virtualization, as mediation in micro process and interoperability in both micro and macro process are all crucial for virtualization and can only achieved by the micro process component foundation.

Service virtualization requires industry to follow a common framework and common practice, so the service providers can provision their services to service consumers by sharing same protocol, interface, service components and information in a transparent environment. The integration uses a vertical approach within the enterprise, and a horizontal approach across the enterprise and the industry. Enterprise interoperability consolidates many different methodologies, systems, terminologies and varieties of style of architecture and integration by using both micro and macro approaches in the vertical direction. There are three steps in achieving the goal. Initially, an ontology needs to take all heterogeneous designs to work on their interference and consolidation into a main route. Second, there is the need for a framework and roadmap to build the foundation of interoperability, and a multi-dimensional mechanism must be implemented to handle interoperability for both micro and macro processes to integrate the assets within the ontology. Third, the integration and tuning between theoretical research and common industry practices to minimize the gap between our strategy and the industry, is the realization step in achieving service virtualization.

In enterprise service provisioning and virtualization, we constructed the skeleton of enterprise service with leverage to each layer and interoperability of the layers. Service virtualization is a process of leveraging enterprise integration strategy within a specific geographical region or across regions. This can be an industry standard scenario such as a credit card transaction in which the service, geography and systems are fully transparent to both cardholders and merchants. A full virtualization environment needs to be built on top of the foundation of component design and service transformation process in order to apply the vertical enterprise integration approach to this final stage, which is the goal of our research work.

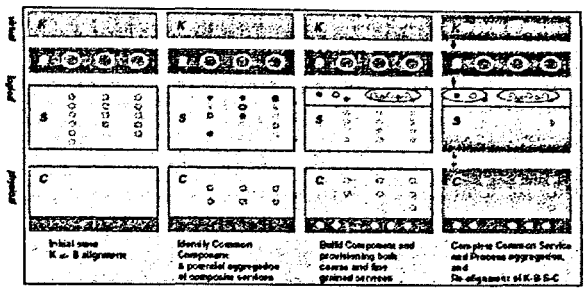


Figure 1. K-B-S-C interoperability and alignment

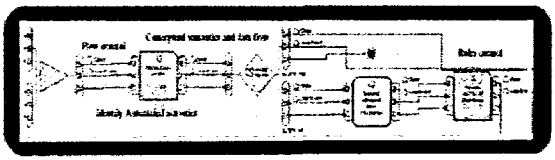


Figure 2. SCA (Service component architecture) of service component

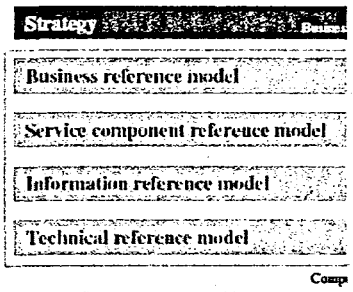


Figure 3. Strategy in interoperabili

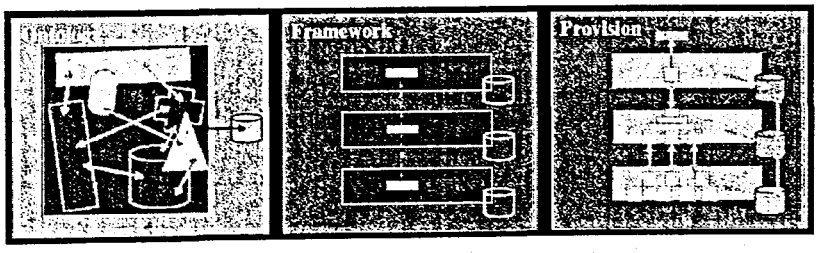


Figure 4. Steps in interoperability approach

Reference

1. R. Barcia and J. Brent, (2005), "Building SOA solutions with the Service Component Architecture", IBM Research
2. L. de Alfaro and T. A. Henzinger, "Interface theories for component-based design," in *Proceedings of the First International Workshop on Embedded Software* (EMSOFT), Lecture Notes in Computer Science 2211,Springer-Verlag, 2001, pp. 148-165.
3. D Ronan et al, (2005), "Web Service Modeling Ontology (WSMO)", European Commission under the projects DIP 2005
4. B. Hutchison and M. Schmidt and D. Wolfson, (2005), "SOA programming model for implementing Web services", IBM Research 2005
5. Jie Lu, Guangquan Zhang & Fengjie Wu, (2005), "Web-based Multi-Criteria Group Decision Support System with Linguistic Term Processing Function", IEEE Intelligent Informatics Bulletin June 2005 Vol.5 No.1
6. Sergey Melnik, (1999), "Generic Interoperability Framework", Distributed and Parallel Databases Journal 1999

A4

A4: Enterprise Integration in eGovernment industry

Publication: Inderscience EG Journal
ISSN (Online): 1740-7508 ISSN (Print): 1740-7494
http://www.inderscience.com/search/index.php?action=record&rec_id=13982

Publication Contact: Prof. Binshan Lin
Editor in chief, EG Journal Inderscience
BellSouth Corporation Professor
College of Business Administration,
BE321, Louisiana State University
Binshan.Lin@lsus.edu



From: Lin, Binshan
Sent: Mon 10/23/2006 7:24 PM
To: Raymond Wu
Subject: EG5349 Paper Submission

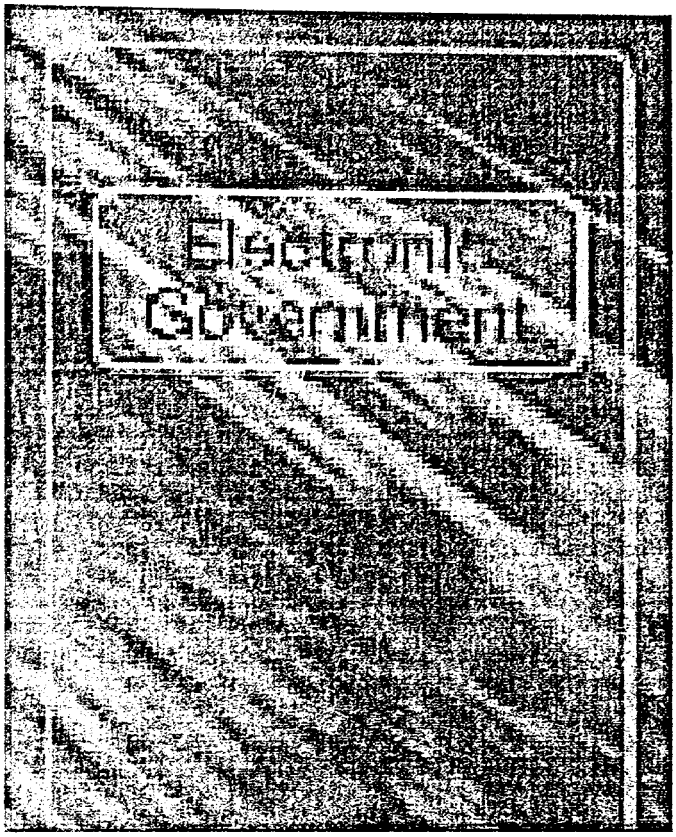
October 23, 2006

Dear Raymond Wu,

After a thorough double-blind review, I am very pleased to inform you that your revised manuscript entitled "Enterprise Integration in e-Government" (EG5349) is scheduled for publication in a forthcoming issue the *Electronic Government: An International Journal* (ISSN: Print 1740-7494, Online 1740-7508) journal, only if you completely fulfill the following seven requirements.

Binshan

Professor Dr. Binshan Lin
BellSouth Corporation Professor
Editor-in-Chief, *Electronic Government: An International Journal* www.inderscience.com/eg



URL: http://www.inderscience.com/search/index.php?action=record&rec_id=13982

Electronic Government, an International Journal (EG)

Volume 4 - Issue 2 - 2007

Table of Contents

Pages	Title and authors
127 - 141	An introduction to e-government for cultural goods in Slovenia *Dejan Lipovšek, Stefan Bojnec*
142 - 166	E-safety: academic issues in occupational safety *Alan C. Smith*
167 - 178	Technology, society and e-government: in search of an ontological framework *Sherif as-Saber, Khalid Hossain, Abdush Srivastava*
179 - 190	**Enterprise integration in e-government** *Raymond Wu*
191 - 203	Information systems in public knowledge management *Peter Gottschalk*
204 - 226	Evaluation instrument for e-government websites *Anders Hedman, Yixin Yi, Belinda Frost, Michael Milutinov*
227 - 246	Smart card implementation in Bahrain: public acceptance and readiness *Adel Ismail Al-Alawi, Mohammed Ameen Al-Amri*
Pages	Title and authors

Enterprise integration in e-government

Raymond Wu

IBM Germany
Schoenaicher Str. 220
Boelingen 71032, Germany
E-mail: rwu_academic@yahoo.com

Abstract: This paper investigates the e-government architecture issue as a means of addressing the interoperability gap. It will incorporate the new knowledge acquired in the e-government integration methodology of the interconnectivity of cross-layers. These layers include the vertical direction of strategy, business, process, service and information. Enterprise architecture mainly mediates between business and IT to minimise the gap by improving governance, agility and business integrity. All of these disciplines and principles should be applied to attain e-government transformation.

Keywords: e-government; integration; metadata; semantic; interoperability; enterprise; architecture; service; virtualisation.

Reference to this paper should be made as follows: Wu, R. (2007) 'Enterprise integration in e-government', *Electronic Government, An International Journal*, Vol. 4, No. 2, pp.179–190.

Biographical notes: Raymond Wu is a Senior IT Architect who has been working with IBM, Oracle, Citibank and Vodafone in the past ten years. His research interests include enterprise architecture and integration, service virtualisation and semantic metadata.

1 Introduction

E-government architecture became pervasive in the 21st century because of its significant growth in terms of huge volume transactions, the citizens' new service concept, and sophisticated businesses. We all understand that in this new, large-scale computing age, high complexity and dynamic development, a common weakness in the interoperability between business and IT is leaving IT far behind in the light of the direction business is taking. Poor business responsiveness and current IT governance make it even more difficult for e-government to achieve the enterprise goal. We identify this overall tactical issue emerging from the vertical gap as the 'invisible enterprise issue' within e-government architecture. Finding new methods to minimise the gap between business and IT by using vertical enterprise integration is the purpose of this paper.

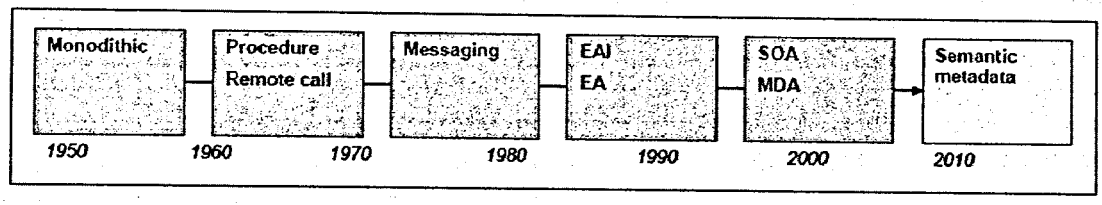
In this century, e-government transformation is one of the biggest challenges within the IT-related sector from the perspective of scale and complexity. The main objective is to adapt existing e-governments to new computing requirements based on the citizens' new service concept of citizen-centric services (Kolsaker and Lee-Kelley, 2006). Thus, increasing service-level features and quality, as well as maintaining a high

Copyright © 2007 Inderscience Enterprises Ltd.

consistency of policies and rules, interoperability and information sharing, productivity and efficiency in addition to reducing operating expenses are all priorities in this transformation. This will require multidimensional aspects from business innovation, service strategy, solution architecture, implementation and integration (Yao, 2005).

Figure 1 shows the evolution of the challenge: our goal is to lessen and counteract this challenge. In the past five years, Service-Oriented (SOA), Model-Driven (MDA), Event-Driven (EDA) and many other leading technologies have been launched for the purpose of solving related problems. Their solutions have already given us significant improvements in e-government; however, the gap is still too massive to cope with the challenges of large scale and high complexity over the next ten years.

Figure 1 The IT evolution and requirements in new age



Such a transformation is crucial as the service concept is changing rapidly: the coherence of the roadmap in e-government transformation is critical as most of the key service providers and vendors still lack a meta-enterprise vision in moving silo into a global roadmap.

The assumption from the discussion to this point is that the e-government issue is not merely caused by enterprise architecture misalignment between business and IT, but that a special style of architecture is needed to cover non-technical areas in order to streamline business implementation.

2 Methodology

The methodology used in this research as it progressed was component base-driven and involved the exploration of vertical e-government integration and common service interoperability between business and IT. The realisation of this methodology first required a technical foundation setup followed by a business semantics study and a new concept of enterprise integration.

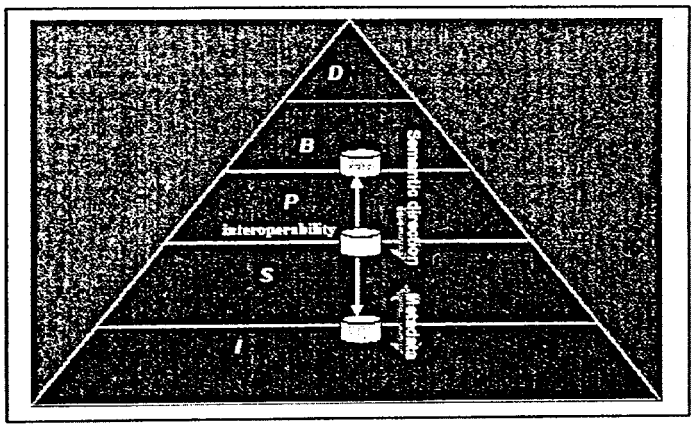
The integrated solution of coherence of approaches forms the basis of this new concept; it serves as the backbone in vertical integration and addresses the e-government enterprise gap. It is expected to provide further insights into micro process integration across e-government enterprise layers.

This integration research was focused on the existing e-government issue, which is the weak interoperability and misalignment between business and IT. The research included an investigation of the root issues, the impact analysis, the nature of interference between business and IT, a conceptual resolution and ended with a strategic direction. The methodologies used in this paper included component base-driven, vertical integration and common services. The proposed roadmap for e-government transformation was planned as follows:

First, in the pre-build stage – decompose the business block into a business component; move forward to a service component through the business modelling process: the reason componentisation needed to be set first is that visibility and traceability (Lytras, 2006) are pre-requirements before vertical integration and common services, *i.e.*, the new concept of interoperability needed to be a robust communication through integration and services; the concept of communication within e-government interoperability needed a service layer across a meta-level component rather than system-to-system or data-to-data integration. In this step, we therefore used componentisation to build up the foundation for integration and services.

Second, in the building stage – after the realisation of componentisation – the other new concept, vertical integration, was launched to work on the interoperability and fragmentation issues, which are the crucial issues in e-government. The integrated solution of coherence of approaches forms the basis of this new concept that acts as a backbone in a vertical direction and addresses the e-government enterprise gap; it is expected to provide further insights into micro process integration across e-government enterprise layers. The three different types of repositories (metadata, services and semantics) collaborate with each other; the strategy of 'enterprise vertical integration' is based on the processes that build up these repositories in the early stage, as well as the methods for achieving interoperability between repositories. The symbolic meta-level repository integration is shown in Figure 2 and clearly demonstrates how the repositories can be integrated within an enterprise architecture environment.

Figure 2 Conceptual ontology interoperability between layers by using 'semantic injection' and 'metadata'



Third, in the post-build and service provisioning stage, common services and common information are the values gained from the vertical integration across organisations; thus common services can build a framework from a virtual infrastructure for an organisation to 'plug and play' if the organisation wishes to save IT cost and follow 100% business compliance. An organisation also has the option to start from an earlier stage and use the service specification to develop its component within this common framework. As to the common information, this mainly consists of data model, meta-level specification, and the data access and information format. The disciplines are not limited only to the data itself, the interoperability between service and data are the key to provisioning common service and information infrastructure.

3 Discussion

Among the e-government challenges, interoperability has been identified as the most crucial. This is because the high integrity of non-system-related policy and rules need to be followed across many geographical entities. Thus a high semantic requirement, robust service interfaces and coherence of enterprise strategy are all important. Moreover, time to services for those mission critical, single source of truth of citizen-based data and e-government interconnectivity to other service industries are all critical; consequently, these requirements reinforce the need for a fundamental change of e-government. These called-for changes can be summarised as follows:

- Componentisation, as mentioned above, became a trend in logical design to cope with the citizens' new service concept and mission critical requirement; these either newly added requirements or existing legacy are moving forward to a component base architecture. There is a need to proceed through the processes of identification, specification and validation (Butler *et al.*, 2005).
- Global-local architecture is the other trend in building up a topology foundation to achieve a common context, although the local authority may add their special rule leverage to global common practices. Within this topology, the gap between organisations is eliminated, as once service is componentised and the concept of government entity is remapped to this new topology, the government organisation is virtualised. At this juncture, development becomes faster as common services provide a pre-build technology foundation. Indeed, the fact that all of the organisations share the same service means they are all managed under the same rules, which were developed using the top-down approach. As all of the organisations share a common protocol to communicate their semantic rules, the integration effort can be eliminated and interoperability can be enhanced through this topology once the semantic communication has been fully developed (Wagner *et al.*, 2006; Moynihan and Batson, 2006).
- Meta-level object definition and single source of truth are priorities in building up an efficient e-government architecture as the integration trend in IT has been changed from peer-to-peer (n:n) to hub and spoke (1:n), and from hub and spoke to service bus (1:1). The pre-requirements for this are the consolidation of all duplicate object and data fragmentation by means of aggregation, merge or summary towards achieving a meta-level. Thus interoperability can be attained within the meta-layer to impact the individual entity effectively (Batini *et al.*, 2006).

When we take the top-down approach into consideration, the strategic layer (*i.e.*, an EU organisation) has the authority to validate the rule within a physical domain and implement the business service through domain analysis: the process is completed by the decomposition of the context into a semantic candidate list. Current UK government guidelines provide a metadata standard, which can be a good reference in producing a candidate list for further micro process integration and interconnectivity between components.

Micro process integration was initiated by the domain context analysis process. In their domain analysis, Lee and Lee (2006) state that once sets of use case scenarios are formulated, the next step will be the extraction of verbs and nouns for identifying domain

objects. The domain analysis concept was applied in supporting the theory of vertical integration in semantic analysis. When combined with the extraction of nouns and verbs, domain analysis make up the common components in process, object and interface as domain assets (Lee and Lee, 2006).

The pre-build semantic repository utilises business domain analysis output in further development. Typical semantic development includes a list of candidate semantic objects. Thus, in addition to the semantic content, meta-level choreography of the semantic interface between different physical domains is also a concern. The semantic interface stored in the service repository needs robust communication with the information metadata repository in order to streamline cross-layer implementation. Therefore, interoperability capability in a vertical direction allows mediation computing entities, such as messaging carrier and class mapping (*i.e.*, XML and JDBC) choreography between service and information to become easier and lead towards a seamless vertical integration (Valkeapää and Hyvönen, 2006).

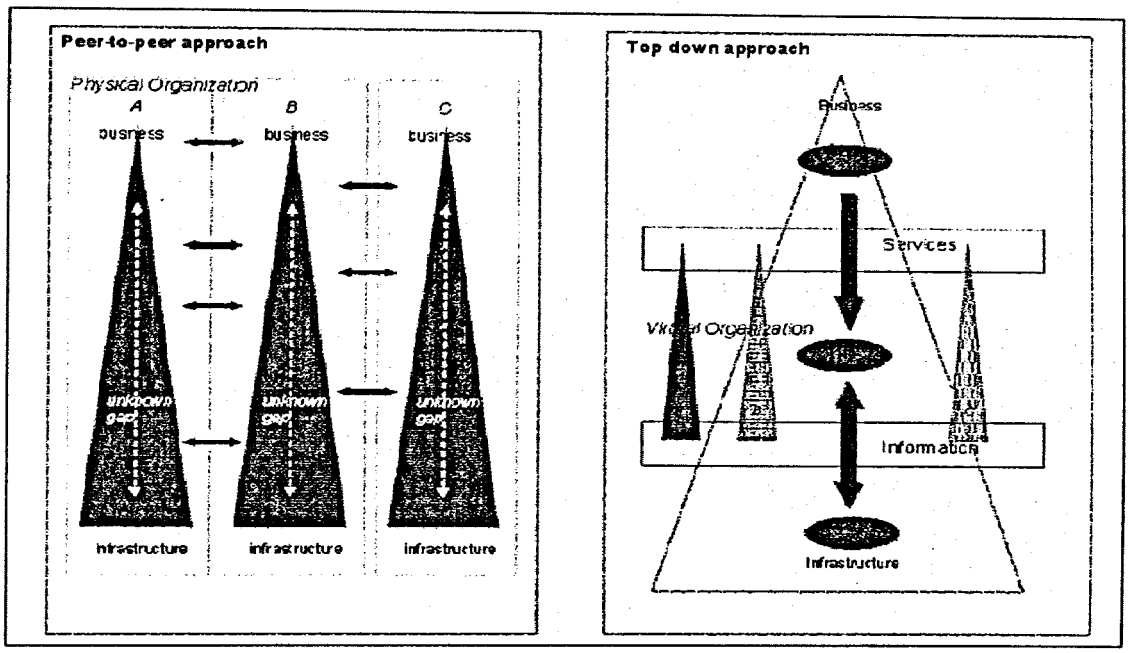
Taking as an example its application in the citizen smart card scenario, we see that 'create a citizen smart card account' is a business service that consists of four composite services – 'citizen medical history', 'security clearance', 'banking records' and 'common application interface'. If we take 'banking records' as an example for further decomposition to atomic services, we see that these include 'loan request', 'channel processing', 'credit and balance check', 'floor limit and fraud check', 'provision service and billing profile', and 'credit grant and adjust floor limit'. All of these services can be analysed for their semantic objects. The use case analysis generates a descriptive sentence of these composite services, which reads as follows: 'a citizen *submits a loan request* for housing' (through form or web): 'administrator *processes application*'; 'authority *processes check and credit review*'; and 'authority *grants and adjusts credit*'. The semantic analysis extracts nouns and verbs from atomic services and forms lists such as this. The semantic repository noun list stores objects such as citizen name, id, phone, account number and an attribute object such as status, credit, authority, and balance while the verb list stores check (balance, credit, floor limit); provision (service profile, billing profile); grant (credit [level 1,2,3]; authority [level 1,2,3]), *etc*. The metadata repository requires a pre-build data schema, data model, symbolic picture of data aggregation and business intelligence to transform the high human-involved development effort to simple configuration. As stated above, the realisation of this concept is through interoperability between the service and metadata repository (Tannenbaum, 2001).

As mentioned above, common service and common information constitute the values proposed through the integration of semantic, service and metadata repositories under a global-local topology; this means that each individual organisation no longer needs to create from scratch. Rather it can adapt its individual entity to the best means, within this common service and information conduit, to build the architecture from the portfolio level and communicate with each other by using a common service approach. The concept of virtual organisation is to prevent any binding or constraint and to adapt the government organisation to a shared common service and information based on demand. This feature enables a flexible topology to provide fast development options for 'plug-and-play'.

Common service and common information is a top-down approach that follows the nature of business implementation direction. As seen in Figure 3, from the technical perspective, the idea was to eliminate the integration effort by using the

'virtual organisation' concept. From the organisation perspective, this enhances the flexibility of service provisioning and the agility between organisation and service: thus, the interoperability across organisations is increased. The realisation of this top-down, vertical integration approach requires robust middleware to support the intercommunication between component and component, and component and repository. In other words, the concept needs physical support from an integration service bus such as the middleware at run-time services.

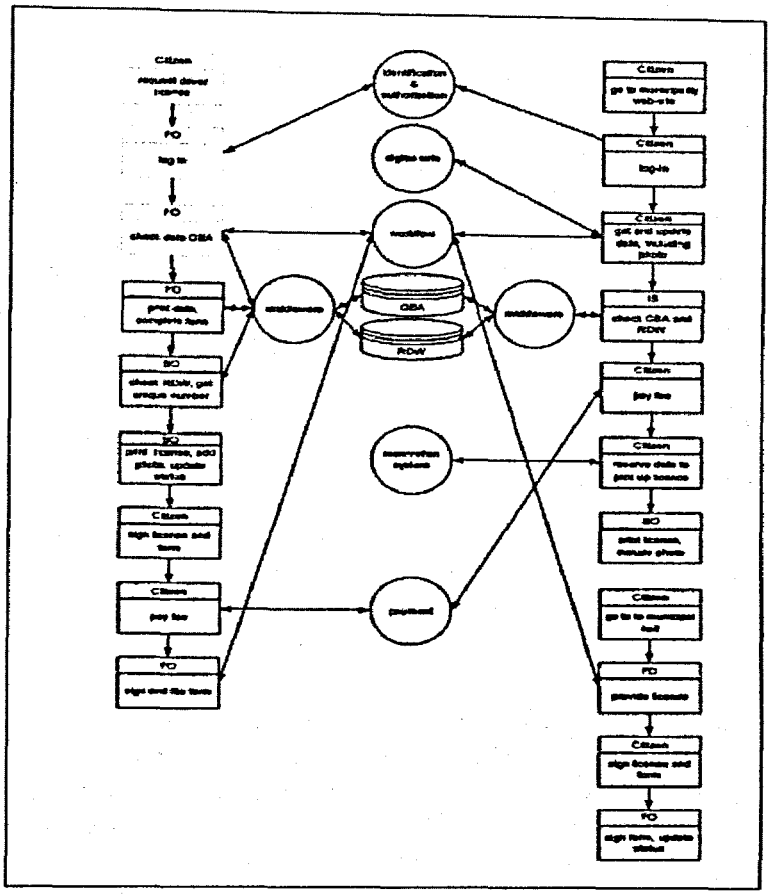
Figure 3 Organisation integration transformation through common services and information



Janssen and Wagenaar (2003), in their "Towards a Flexible ICT-Architecture for Multi-Channel E-Government Service Provisioning", gave the following business scenario: a citizen goes to the counter of the town hall and requests a (new) driver's licence (refer to Figure 4): the process requires reengineering and design guidelines, which can be summarised as:

- information should be captured only once at the source and re-used by other components
- communication between components should be kept to a minimum
- there should be a central process-control component integrating business process steps with functionality provided by the components
- components should be able to communicate with each other directly or through middleware.

Figure 4 Business scenario (citizen's application for driver's licence)



From the previous design guidelines, the assumption can be made that citizen data integrity, componentisation and semantic centric implementation, service interface and business process within municipality systems are all keys to e-government transformation. This implies that tactical e-government issues, including data fragmentation, duplicate process and silo, multipoints integration and inefficient use cases are all within the scope of the top-down approach. This approach consists of componentisation, vertical integration, common services and semantic implementation (which is still new in e-government architecture and calls for further investigation). The proposition is that the following procedures need to be implemented before moving from silo to global mainstream: first – the semantic metadata format and protocol need be unified and synchronised in relevant geographical locations; second – terminology, natural language, semantic attribute and the access method need to be standardised in order to eliminate the semantic communication gap between two semantic groups.

In addition to semantic implementation, a service interface needs to be established to reach citizens. Thus, before service can be reached through channel and pervasive technology, the information foundation needs to first be consolidated for the purpose of building up a master data identified as 'single source of truth'. This should be established through data identification, consolidation, cleansing and validation processes so that

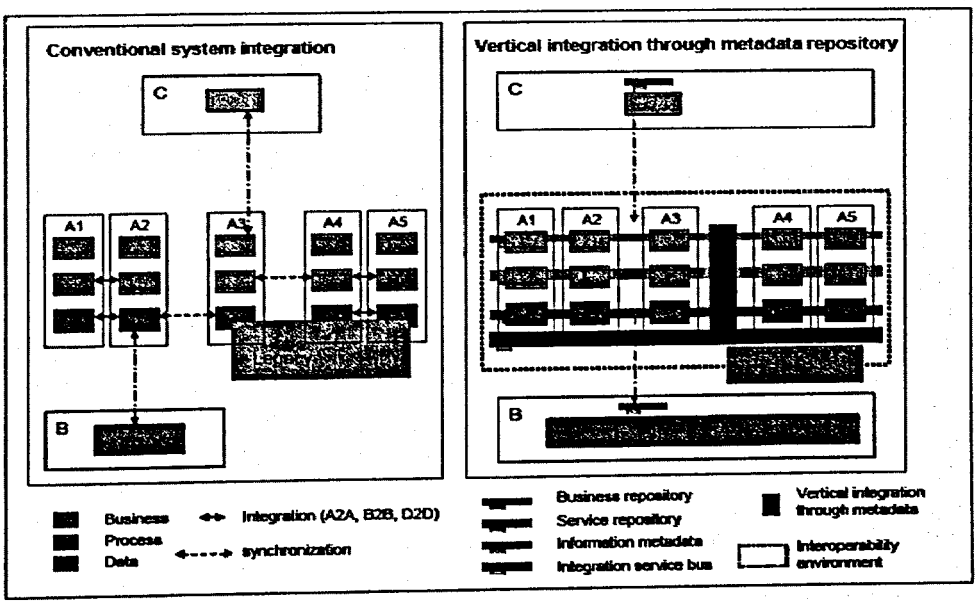
information can follow both a global-local structure and principles such as single point data manipulation and channel communication elimination, *etc*. Once a citizens' master data is built, service intelligence can be implemented to add value through service consumer insight and provisioning macro-trend analysis and personalisation. Once in place, this foundation would provide an efficient interface to reach citizens through channel and pervasive technology.

To expand the service flexibility across organisations, a virtual organisation concept was introduced as mentioned above. The e-government virtual organisation presented in this paper is intended to highlight the service establishment through a combination of services provisioned by virtual organisations that leverage the policies, regulations, services and decisions into a single point within the vertical skeleton to facilitate cross-organisation services and overcome time and space limitations. In contrast to system integration, top-down vertical integration starts from context and is followed by service and information metadata to facilitate vertical-integrated solutions and provide citizens with common service and common information. Figure 5 shows the integration of three repositories in global-local topology. The cross-organisation integration known as integration service bus is seen as providing a physical connectivity service, such as middleware.

Elements of virtual organisation include the following:

- the relationship of a virtual organisation is based on a particular competency that complements the others
- it capitalises on the mobility and responsiveness of telecommunications to overcome problems of distance
- timing is a key aspect of relationships, with individuals using responsiveness and availability to decide between alternatives
- there must be trust between individuals separated in space for virtual organisation to be effective (Scott, 1999).

Figure 5 Differentiators of vertical integration from conventional integration



Virtual organisation can be distinguished from hierarchical and network forms of organisation by its rejection of status boundaries and the lack of importance it ascribes to proximity. Formal hierarchies may connect distant offices and plants in a flow of information and material resources, but they enforce distinctions between departments as part of a system of top-down command and control (Bush and Frohman, 1991) and rely on the proximity of management and workers within departments and offices to develop the trust necessary for a functional system (Handy, 1995; Bush and Frohman, 1991). This relationship and trust can be established under a top-down approach by using global-local topology and vertical integration as opposed to peer-to-peer integration. The latter is limited in supporting relationships for the formation of a virtual organisation, which requires a particular competency within a group, IT loss mobility, and responsiveness and timing from the complexity and agility perspective. For this reason, peer-to-peer integration within e-government cannot be an asset from the virtual organisation point of view. From the common industry methodology perspective, it is not difficult to discern that the concept of vertical integration was already in a prototypical stage although the trend had not yet been specified. The evidence for early signs of this new concept may be seen in the following:

- Zachman was almost the frontier of the enterprise framework: it never focused on any one architecture layer only, rather it introduced cross-mapping between the dimensional matrixes and was considered the first indication that vertical integration existed.
- Rational Unify Process (RUP) was used as a methodology of abstraction in identifying a boundary to look at use case in each small derivation and iteration and guided its architecture development in a vertical direction.
- SOA and Model-Driven Architecture (MDA) were architecture styles providing further indication that 'loosely coupled' was actually a vertical alignment of business and IT as a foundation for the development of further business lines.

D'Souza (2001) defines vertical integration in his "MDA (Model-Driven Architecture) and Integration Opportunities and Challenges" and points out that vertical integration came from the different levels of abstraction within the same system, physical data models, logical data models, networks, application specifications, component assemblies, business process models, and business goals and strategies.

This viewpoint provides further evidence for the feasibility of the top-down vertical integration. From the nature of the business implementation direction we have several options for the enhancement of interoperability. These include A2A (application to application); B2B (business to business); D2D (data to data); or vertical B2A2D (business to application to data). If we start from A2A or D2D, the risk is much higher than B2B because reverse engineering in Business Process Reengineering (BPR) requires both upward and downward reengineering and n:n repeat integration efforts for both cases; this can cause a high risk if the change started from the business layer and each system was developed in a different scenario. B2B has lower risk concerns but is weaker owing to its having a lower sharing model and component. By comparison, B2A2D can have better business compliance across organisations with higher sharing common components, services and lower risk; however, the technologies cover many advanced areas such as semantic transmission, grid concept, componentisation and service

choreography, *etc.*, which need further review. For example, if we assume that we have most of the technology ready for this interoperability by using vertical integration and a top-down approach, then even a minor weakness in the componentisation of a business area that needs to apply a new policy change or organisational restructuring, with the potential for creating a big gap from 'business change' to 'common service' within this area, can cause delay or defect. The most likely reason for these setbacks would be that the method or componentisation tool is not mature enough to achieve a full 'separation of concern' level. Another area of concern would be the mapping between the semantic, service and metadata. Thus, bringing broad technologies and concept coherence to full realisation needs further meticulous research.

Layne and Lee (2001) have the viewpoint that e-government provides progress towards a higher level of integration and interoperability to government levels and branches. In essence, interoperability leads to extensive information sharing among and between governmental entities. However, the obstacles preventing rapid progress in that direction are not merely technical. In fact, the technology side may prove the least difficult to address; the organisational, legal, political and social aspects may prove to be much more of a challenge (Layne and Lee, 2001).

This finding implies that the first steps towards interoperability cannot be seen merely from technical aspects such as A2A, D2D or componentisation of SOA. Rather it can be a high risk from a recursive reengineering requirement through each business change caused by either government organisation or political issue. Indeed, the political and organisational aspects are more challenging areas than technology. The maturity of the vertical integration concept and provisioning of cross-organisation share service and information have a better chance of coping with both the political and organisation issues and in eliminating technical integration complexity. This is evident from the nature of business direction, virtualisation and interoperability perspectives (Liu, 2005).

4 Conclusion

The concept of vertical integration in e-government is achieved by applying semantic, service and metadata repositories to a global-local architecture foundation. The top-down, vertical integration methodology follows the nature of business implementation direction and has a better chance of coping with interoperability requirements. Basically, it can eliminate the point-to-point integration effort and risk, as well as provide a common service and common information value proposition, such as portfolio infrastructure for quick development and policy compliance.

The nature of 'e-government vertical integration' distinguishes itself from traditional integration by effective transmission from business context and semantics into common components to produce 'run-time' services and communicate with metadata through the micro-componentisation process in a development environment.

The lack of interconnectivity among context, semantics, service and data are the main concerns with the implication that e-government architecture needs a robust micro-mechanism of semantic messaging and metadata to coordinate across layers. In this way the strategic direction will have an opportunity to impact the business process to streamline successful services and information components.

References

Batini, C., Grosso, R. and Longobardi, G. (2006) 'Design of repositories of conceptual schemas for large-scale e-government projects', *Electronic Government, An International Journal*, Vol. 3, No. 3, pp.306–328.

Bush, J., Jr. and Frohman, A. (1991) 'Communication in a "network" organization', *Organizational Dynamics*, Vol. 20, No. 2, pp.23–36.

Butler, J., Mayo, D. and Weiler, J. (2005) *Succeeding with Component-based Architecture in e-Government*, FEA-PMO.

DSouza, D. (2001) *Model-Driven Architecture and Integration Opportunities and Challenges*, Kinetium.

Handy, C. (1995) 'Trust and the virtual organization', *Harvard Business Review*, Vol. 73, No. 3, pp.40–50.

Janssen, M. and Wagenaar, R. (2003) 'Towards a flexible ICT-architecture for multi-channel e-government service provisioning', *IEEE Computer Society, Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)*.

Kolsaker, A. and Lee-Kelley, L. (2006) 'Citizen-centric e-government: a critique of the UK Model', *Electronic Government, An International Journal*, Vol. 3, No. 2, pp.127–138.

Layne, K. and Lee, J. (2001) 'Developing fully functional e-government: a four-stage model', *Government Information Quarterly*, Vol. 18, pp.122–136.

Lee, H. and Lee, J. (2006) *Analyzing Business Domain: A Methodology and Repository System*, Korea Advanced Inst. of S&T.

Liu, C. (2005) 'Using e-governmental indicators to build virtual value chain', *Electronic Government, An International Journal*, Vol. 2, No. 3, pp.277–291.

Lytras, M. (2006) 'The semantic electronic government: knowledge management for citizen relationship and new assessment scenarios', *Electronic Government, An International Journal*, Vol. 3, No. 1, pp.5–17.

Moynihan, G. and Batson, R. (2006) 'Development of a matrix methodology for database reengineering and improvement: an e-government case study', *Electronic Government, An International Journal*, Vol. 3, No. 2, pp.190–203.

Scott, M. (1999) 'Virtual organization as process: integrating cognitive and social structure across time and space', *Journal of Computer-Mediated Communication and Organization Science*.

Tannenbaum, A. (2001) *Using Metamodels, Repositories, XML, and Enterprise Portals to Generate Information on Demand*, Boston, MA: Addison Wesley Professional.

Valkeapää, O. and Hyvönen, E. (2006) 'Semantic annotation with browser-based annotation tool', *1st Asian Semantic Web Conference*.

Wagner, C., Cheung, K., Ip, R. and Bottcher, S. (2006) 'Building semantic webs for e-government with Wiki technology', *Electronic Government, An International Journal*, Vol. 3, No. 1, pp.36–55.

Yao, W. (2005) 'Interoperability: an indispensable propellant of e-China', *International Journal of Services and Standards*, Vol. 1, No. 4, pp.446–452.

Bibliography

Foster, C. and Stentiford, P. (1999) 'Metadata – the key to content management services, proceedings of the meta-data', *The Third IEEE Meta-Data Conference*.

Lu, J., Zhang, G. and Wu, F. (2005) *Web-based Multi-criteria Group Decision Support System with Linguistic Term Processing Function*, IEEE Intelligent Informatics, Vol. 5, No. 1.

Staab, M., Stojanovic, N., Studer, R. and Sure, Y. (2001) *Semantic Portal – The SEAL Approach. To Appear in: Creating the Semantic Web*, MIT Press.

Vehviläinen, A.I., Alm, O. and Hyvönen, E. (2006) 'Combining case-based reasoning and semantic indexing in a question-answer service', Poster paper, *1st Asian Semantic Web Conference*, 20 June.

Wu, R. (2006) 'Enterprise vertical integration in semantic environment', *IBFR GCBF Conference*.

A5: Enterprise Component based Interoperability in e-Government

Publication: Haamaha July 2007

International Conference on Human Aspect in Advanced Manufacturing and Hybrid Automation

9th-12th July, 2007, Poznan, Poland (http://www.ergonet.net/haamaha2007)

Publication Contact: Prof. Stefan Trzcieli

Chairman, Haamaha 2007

Vice Dean of Computing and Mgt

Poznan University of Technology,

Stefan.Trzcielinski@put.poznan.pl



From: "HAAMAHA 2007" <haamaha@put.poznan.pl>

To: rwu_academic@yahoo.com

Subject: Re: HA5K018

Date: Wed, 7 Mar 2007 19:16:34 +0100

Confirmation and invitation for participation at International Conference on Human Aspect in Advanced Manufacturing and Hybrid Automation " Managing Enterprise of the Future " July 9 th – 12 th , 2007. Poznan University of Technology ,

Dear Mr RAYMOND WU ,

RE: HA5K018 Enterprise Component based interoperability in e-Government

Congratulations! I am pleased to inform you that your paper(s) has been accepted for presentation at HAAMAHA ' 07. If you have co-authors whose e-mail addresses were not provided earlier, Poznan , Poland.

Sincerely Yours,

Stefan Trzcielinski

Stefan Trzcieliński, Ph.D., D.Sc. Associate Professor Vice Dean of Faculty of Computing and Management Poznan University of Technology, Institute of Management Engineering

trieval system
ermission from

Otganizations in Changing Environment. Current Problems,Concepts and Methods of Management

Proceedings of the Eleventh International Conference on Human Aspects of Advanced Manufacturing: Agility and Hybrid Automation
4th International Conference ERGON-AXIA

HAAMAHA 2007

Managing Enterprise of the Future

July 9-12, 2007 Poznań, Poland

Edited by:

Wiesław M. Grudzewski
Irena Hejduk
Institute of Organization and Management in Industry

Stefan Trzcieliński Poznań
University of Technology

2007

IEA Press International Ergonomics
Association 510 Walnut Street 575
WARF Madison, WI 53726 USA

o persons or
lse or
operatech.

Modern Enterprise

1 A framework of the work service for future workers Jung Jong Myung, Yun JungHyun, Mo Kyoung Seon Lee Dong Myun, Kim Jung Ha
2 The impact of IT solutions on firm performance
reserch on companies listed at the JSX1
Sadowska Dorota, Tutak Sabrina, Stalmaska-Szyc Dorota Yubi
3 Influence of genetic algorithms on development on models of production organization
Dudzik Jarosław, Zawiła-Niedźwiecki
4 Macro process design in component based architecture
Wu Raymond
5 Enterprise architecture evolution components
Nowak Lech, Nowak Jacek
6 Dynamic process modeling in component based integration
Wu Raymond
7 The role of architecture evaluations in ICT companies Hämäläinen Niina, Ylikoski Tuija, Niemi Eetu
8 Enterprise architecture process of a telecommunication company a case study on initialization
Ariadne in An, Hämäläinen Niina
9 Design and information technologies
tools for innovation's processes in Mexico
Meza Aguilar Jorge
10 Enterprise component based interoperability in e-government
Wu Raymond
11 Intelligent agents for supply chain business applications
Olejnik Joanna, Iwaszkiewicz Marek
12 Complex biometric algorithms for people identification
Dębowski Adam, Mucha Jan, Szymański Piotr
13 The impact of web services on companies
Hassan Tarawneh, Djoni Hartono M Noor Brandjida, Sanwa Gaeby, Mohamed Bakaki
14 The internet in polish small and medium sized enterprises Aneżka Teresa Annotating multimedia documents with multimedia documents in a decision support system Charles Robert nhlimo automating work time losses

Enterprise Component based interoperability in e-Government

Raymond Wu
IBM Software Group
+49 7031 16 1755 / raymond@de.ibm.com

We reviewed our recent work in component based implementation and use example in e-Government to present the design process from micro component design to macro component based service virtualization. It demonstrate how services are derived from business components, and how they are designed and produced in achieving Service virtualization. e-Government rely on new services to mediate its services virtualization, there are two key processes play crucial roles in this mediation: the micro component design and macro metadata framework, the supplementary solutions of these two processes construct the enterprise integration interoperability and service virtualization.

Keywords
Integration, Metadata, Interoperability, Enterprise, Architecture, Service, e-Government

A typical e-Government project usually consists of domain analysis, architecture and shared service. The domain analysis group is divided into several teams to handle various LOB (line of business) as each team will have different areas of business domain knowledge; each team may in fact handle several business lines (i.e., use cases). It's very often the case that the individual LOB team doesn't have adequate concepts and communication to determine the common components, which include common services, common interface and standard data access and data models, we would like to review and list these common issues in enterprise integration with our proposals of resolution, this is shown in Figure 1.

- Too many integration points cause implementation silos
- Data fragmentation with duplicate process without aggregation strategy [1]
- Common area is invisible and unable to reuse in a shard model
- Lack of priority and focus of strategy slow down business responsiveness
- Weak enterprise interoperability cause high risk in industry transformation

In the recent requirements, we experienced that the public services, when handling the conventional credit card, doesn't maintain a robust "Fraud detection policy" at account level, so many transactions become invalid, the agility and responsiveness of making a proper change is still weak and many consumers suffer the serious issues that their transaction need be rolled back. For this reason, we launched add-on service component of Fraud Policy service to replace the conventional practice and

dynamically retrieve card holder profile and the applied fraud policy to determine the authorization approval through inter-banking, so each account holder can have their security profile and preference for the card issuer or dealers to determine the individual fraud detection based on the transaction behavior and customer profile. The other example shown in Figure. 2, was when the e-Government integrated with other industry such as financial or telecom to provide public services, how to define their interface and how to integrate the emerging technologies into multi-channel service in e-Government is also rely on component based implementation.

M. Janssen and R. Wagenaar, in their "Towards a Flexible ICT-Architecture for Multi-Channel e-Government Service Provisioning," gave the following business scenario: a citizen goes to the counter of the town hall and requests a (new) driver's license (refer to Figure 3); the process requires re-engineering and design guidelines which can be summarized as -- (1) information should be captured only once at the source and re-used by other components; (2) communication between components should be kept to a minimum; (3) there should be a central process-control component integrating business process steps with functionality provided by the components; and (4) components should be able to communicate with each other directly or through middleware (Janssen and Wagenaar 2003). From the above design guidelines, the assumption can be made that citizen data integrity, componentization and centric semantic implementation, service interface and business process within municipality systems are all keys to e-Government transformation. This implies that tactical e-Government issues including data fragmentation, duplicate process and silo, multi-points integration and inefficient use cases are all within the scope of the top down approach. This approach consists of componentization, vertical integration, common services and semantic implementation (which is still new in e-Government architecture and calls for further investigation). The proposition is that the following procedures need to be implemented before moving from silo to global mainstream: first -- the semantic metadata format and protocol need be unified and synchronized in relevant geographical locations; second -- terminology, natural language, semantic attribute and the access method need to be standardized in order to eliminate the semantic communication gap between two semantic groups.

The methodology used in this research as it progressed was component-base driven and involved the exploration of e-Government integration in vertical direction and common service interoperability between business and IT. The realization of this methodology first required a technical foundation setup followed by a business semantics study and a new concept of enterprise integration. The integrated solution of coherence of approaches forms the basis of this new concept; it serves as the backbone in vertical integration and addresses the e-Government enterprise gap. It is expected to provide further insights into micro process integration across e-Government enterprise layers. This integration research was focused on the existing e-Government issue which is the weak interoperability and misalignment between business and IT. The research included an investigation of the root issues, the impact analysis, the nature of interference between business and IT, a conceptual resolution and ended with a strategic direction. The methodologies used in this paper included component-base driven, vertical integration and common services. The proposed roadmap for e-Government transformation was planned as follows:

First, in the pre-build stage - decompose the business block into a business component; move forward to a service component through the business modeling process; the reason componentization needed to be set first because visibility and traceability (Lytras 2006) are pre-requirements before vertical integration and common services, i.e. the new concept of interoperability needed to be a robust communication through integration and services; the concept of communication within

e-Government interoperability needed a service layer across a meta-level component rather than system to system or data to data integration. In this step we therefore used componentization to build up the foundation for integration and services.

Second, in the building stage - after the realization of componentization, the other new concept – vertical integration -- was launched to work on the interoperability and fragmentation issues which are the crucial issues in e-Government. The integrated solution of coherence of approaches forms the basis of this new concept that acts as a backbone in a vertical direction and addresses the e-Government enterprise gap; it is expected to provide further insights into micro process integration across e-Government enterprise layers. The three different types of repositories (metadata, services and semantics) collaborate with each other; the strategy of "enterprise vertical integration" is based on the processes that build up these repositories in the early stage as well as the methods for achieving interoperability between repositories. Third, in the post-build and service provisioning stage - common services and common information are the values gained from the vertical integration across organizations; thus common services can build a framework from a virtual infrastructure for an organization to "plug and play" if the organization wishes to save IT cost and follow 100% business compliance. An organization also has the option to start from an earlier stage and use the service specification to develop its component within this common framework. As to the common information, this mainly consists of data model, meta-level specification, and the data access and information format. The disciplines are not limited only to the data itself, the interoperability between service and data are the key to provisioning common service and information infrastructure.

e-Government is one of the high complexity industry among various sectors. To eliminate the risk, our assumption is; the higher occupancy of common services within enterprise, the less suffering can be caused in each IT and business transformation, as technology and business innovation and transformation create opportunity while it may impact enterprise seriously if the transformation foundation is not solid, when they are enforced to transform their business or IT to be survival in next stage, weakness in enterprise skeleton and interoperability supported by common component may cause high risk in each transformation. According to L. Yu et al., An obstacle to software reuse is the large number of major modifications that frequently have to be made as a consequence of dependencies within the reused software components. Component base implementation is the foundation in coping with this bottleneck [2] [3].

To cope with these issues, we followed the enterprise integration strategy in our recent research which includes, validation of business and service component, service analysis and transformation, service design, information aggregation and service provisioning. Within the enterprise integration life cycle, component validation and analysis, which we introduced in earlier research papers is the foundation in component design and industry service virtualization.

Scott M. Preston (1999) argues that virtual organization adds value and requires a different way of perceiving the world by those who wish to participate in it. Elements of virtual organization include the following: (1) the relationship of a virtual organization is based on a particular competency that complements the others; (2) it capitalizes on the mobility and responsiveness of telecommunications to overcome problems of distance; (3) timing is a key aspect of relationships, with individuals using responsiveness and availability to decide between alternatives; and (4) there must be

trust between individuals separated in space for virtual organization to be effective (Scott 1999) Virtual organization can be distinguished from hierarchical and network forms of organization by its rejection of status boundaries and the lack of importance it ascribes to proximity. Formal hierarchies may connect distant offices and plants in a flow of information and material resources, but they enforce distinctions between departments as part of a system of top down command and control (Bush and Frohman, 1991) and rely on the proximity of management and workers within departments and offices to develop the trust necessary for a functional system (Handy, 1995) (Bush and Frohman 1991) (Handy 1995). This relationship and trust can be established under a top down approach by using global-local topology and vertical integration as opposed to peer-to-peer integration. The latter is limited in supporting relationships for the formation of a virtual organization which requires a particular competency within a group, IT loss mobility, and responsiveness and timing from the complexity and agility perspective. For this reason, peer-to-peer integration within e-Government cannot be an asset from the virtual organization point of view. From the industry common methodology perspective, it is not difficult to discern that the concept of vertical integration was already in a prototypical stage although the trend had not yet been specified.

Layne and Lee (2001) have the viewpoint that e-Government provides progress toward a higher level of integration and interoperability to government levels and branches. In essence, interoperability leads to extensive information sharing among and between governmental entities. However, the obstacles preventing rapid progress in that direction are not merely technical. In fact, the technology side may prove the least difficult to address; the organizational, legal, political and social aspects may prove to be much more of a challenge (Layne and Lee 2001).

Component validation and analysis consists of steps in validation of business components, service components, and physical components, common component parsing analysis, service aggregation parsing analysis, and implementation priority and skeleton. From the technical perspective the idea was to eliminate the integration effort by using the "virtual organization" concept, from the organization perspective, this enhances the flexibility of service provisioning and the agility between organization and service; thus, the interoperability across organizations is increased.

The realization of this top-down vertical integration approach requires robust mediation to support the inter-communication between component-component and component- repository. In other words, the concept needs a macro framework and physical support in micro integration service bus such as the middleware in realization of run-time services. The lack of interconnectivity between context, semantics, service and data are the main concerns with the implication that e-Government architecture needs a robust micro-mechanism of semantic messaging and metadata to coordinate across layers. In this way the strategic direction will have an opportunity to impact the business process to streamline successful services and information components.

The conventional design doesn't have the dynamic control to have customer insight while the issues were resolved in component based design. This is the service transformation and provisioning example which mediate the enterprise components in forming the skeleton. Service components derived from business use case are the mediation between business and technical element which integrate all business semantic, interfaces and technical configuration into a logical component, initially all the service components are in atomic level [4] [5]. In the component validation and

analysis research, we applied group decision model in validation process, so business component with its associated service components are selected and validated through decision. We further detailed the rational model in analyzing common component and aggregated services by using use case parser to determine enterprise component in forming the skeleton of integration. Aggregation is a kind of service normalization in grouping all atomic service element from fine-grained to coarse-grained component.

We also introduced the component design and interconnectivity for how the component communicate each other and how the micro process of business semantic be derived into each component. The component element is the physical element that mediates service and information, the design of service components follows our previous strategy, by using business semantic injection to take the context into the semantic and micro processes, so that each component has its contract level responsibility in provisioning a subset of service, and the interface helps the coordination between components and metadata build the foundation of interoperability within the service layer and between layers.

- At the very beginning of a use case, the first event triggers service Φ_1

  $\varepsilon^1 : r_0 \times \Phi_1 \rightarrow r_1$

- When the service event Φi ($i <> 1$) is read and triggers the service component, we express the state transformation as follows:

- $\varepsilon^i : r_{i-1} \times \Phi_i \rightarrow r_i$, we further derive the transformation as follows:

  $\varepsilon^{i+1} = (r_i) \times \Phi_{i+1} \rightarrow r_{i+1} = (r_{i-1} \times \Phi_i) \times \Phi_{i+1} \rightarrow r_{i+1}$

- Which is in equilibrium to $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$, and derives $\varepsilon^{nm} : r_n \times (\Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m) \rightarrow r_{m\ (n<m)}$, we call this the "semantic event aggregator."

- As the final step of a use case, the final event trigger service Φ_n

  $\varepsilon^n : r_{n-1} \times \Phi_n \rightarrow r_n$, Φ_n is a terminator, an example of which is the settlement of a daily transaction.

- The string $\Phi (r_n , r_m) = \Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m$ is a path from r_n to r_m, we call "service route" and $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$ a "composite service" between r_{i-1} and r_{i+1}, we use symbol $\Phi_{i, i+1}$ to denote composite service

- When frequency $F(\Phi_{i, i+1}) = F(\Phi_i \times \Phi_{i+1})$ is higher than threshold σ, we aggregate the concatenated alphabet (part of the string) $\Phi_i \times \Phi_{i+1}$ into $\Phi_{i , i+1}$, this imply the original state r_i between r_{i-1} and r_{i+1} will no more exist for this composite service.

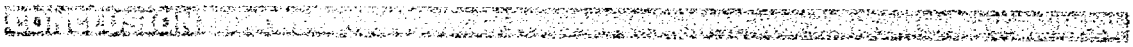
The foundation of this skeleton is interconnectivity between service component by using common component and aggregated component. We further develop the service virtualization on top of this interoperability skeleton, which we call enterprise vertical integration, in extending the integration across industry and geographical areas, this is the virtualization service in horizontal direction, once industry of different regions build up their semantic interoperability in three metadata repositories (business, service and information), they can synchronize their business, operation and strategy, so that the limitation of business, geographical area, products, application, and data structure become transparent, this is what we call service virtualization. Once the vertical skeleton and service extension are built, the interoperability reflected from the business layer has the capability of integrating cross-organization and cross-geographical constraints, as fully transparent services become possible if each system

or organization follows an industry standard by using metadata strategy and an industry framework. There are three steps to reach service virtualization; [6]

a). To identify the unique source of truth within enterprise, and use its metadata within business, service and information as a corporative standard, so when other systems and applications consolidated into enterprise repository, there is a standard and guideline to follow.

b). Vertical enterprise integration, as when other systems consolidated into the enterprise, there will be some metadata out of scope and need further integration effort to streamline the vertical integration.

c.) Service virtualization is the objective of our enterprise integration, we need to make sure our repository holding enough and accurate information within industry standard to integrate enterprise to external.



e-Government is facing a big challenge within enterprise integration as the service concept is moving, global policy and regulation is changing and the high demands in transforming the system silos to an industry and global standard is not an easy task.

The concept of vertical integration in e-Government is achieved by applying semantic, service and metadata repositories to a global-local architecture foundation. The top down vertical integration methodology follows the nature of business implementation direction and has a better chance of coping with interoperability requirements. Basically, it can eliminate the point-to-point integration effort and risk as well as providing a common service and common information value proposition such as portfolio infrastructure for quick development and policy compliance.

We gave the example of financial services in public sector for how to narrow down the root issue within a small scope as a service component while maintaining its alignment to both business and technical components, this service component architecture speed up the change and enhance dramatically the integrity and responsiveness. An example of service provisioning would be the case where several key industry vendors have built up an industry framework for service providers in that particular industry to follow, starting from business domain analysis and working through to component deployment. So each service provider benefits from the industry common services and common interface, particularly those B2B, B2C, merger and acquire cases. Any business activity becomes easier when the various parties involved have a common understanding and consensus on business terms, service definitions, interfaces and component specification, and these become transparent and common practices [7].

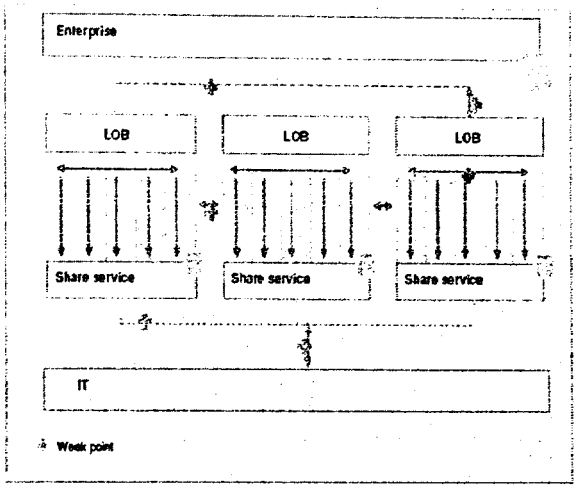


Figure 1. Project architecture common issue

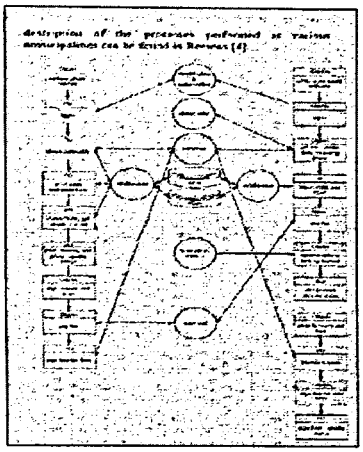


Figure 3– Business scenario (citizen's application for driver license)

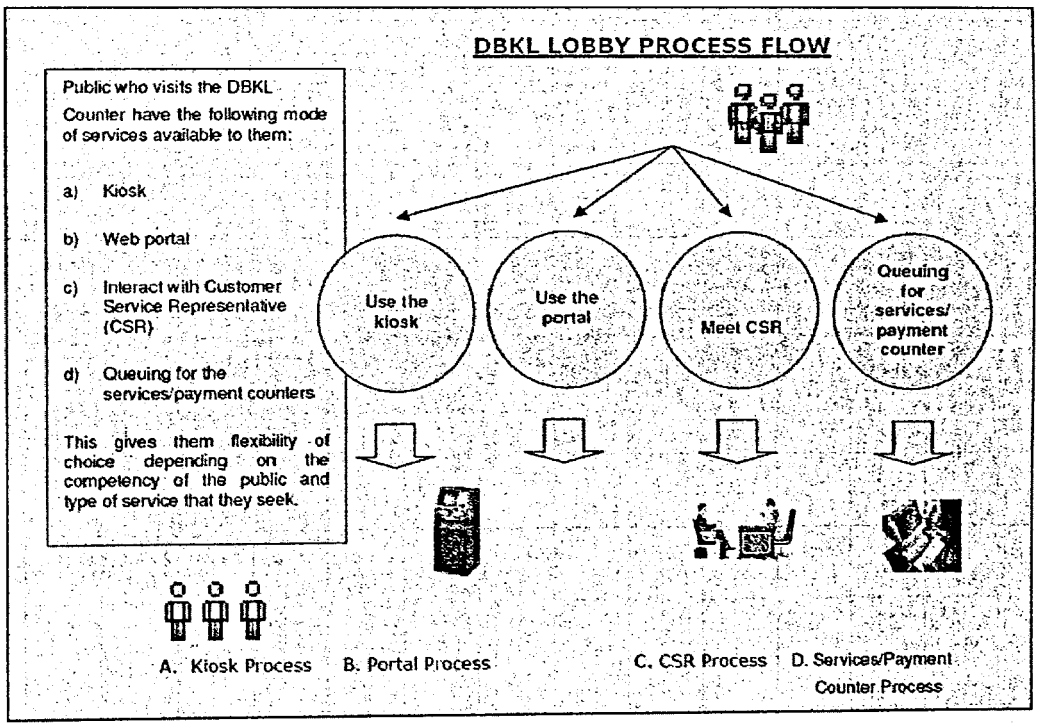


Figure 2. Multi-channel and industry integration in public services

1. Curtis, K., Foster, P. W., Stentiford, F. (1999), "Metadata - The Key to Content Management Services", Proceedings of the Meta-Data '99, The third IEEE Meta-Data conference, April 6-7, 1999, Bethesa, Maryland, USA
2. DSouza,(2001)," Model-Driven Architecture and Integration Opportunities and Challenges", Kinetium
3. L. Yu, (2005), "Categorization of Common Coupling in Kernel-Based Software," 43rd ACM Southeast Conference, 2005
4. M. Staab, N. Stojanovic, R. Studer, and Y. Sure, (2001), "Semantic Portal - The SEAL approach. to appear in: Creating the Semantic Web.", MIT Press
5. O. Valkeapää and E. Hyvönen, (2006), "Semantic Annotation with Browser-based Annotation Tool", 1st Asian Semantic Web Conference
6. J. Hunter , (2001), "MetaNet - A Metadata Term Thesaurus to Enable Semantic Interoperability Between Metadata Domains", Journal of Digital Information, Volume 1 Issue 8
7. O. Zimmermann, P. Krogdahl, C. Gee, (2004), "Elements of Service-Oriented Analysis and Design", IBM Academy

A6

A6: Component Design, Integration and Service Transformation in Financial Industry

Publication: IJISCM (IJISCM-0628) International Journal of Information Systems and Change Management (IJISCM) ISSN (Online): 1479-313X - ISSN (Print): 1479-3121
http://www.inderscience.com/browse/index.php?journalCODE=ijiscm

Publication Contact: Professor David C. Chou
Editor-in-Chief, IJISCM
Department of Computer Information Systems
Eastern Michigan University
Ypsilanti, MI 48197
Tel: 734-487-0054. Fax: 734-487-1941
Email: david.chou@emich.edu

Date: Wed, 06 Jun 2007 14:50:03 -0400

From: "David C Chou" <dchou@emich.edu> Add to Address Book Add Mobile Alert

Subject: Acceptance letter of IJISCM-0628

To: "Raymond Wu" <rwu_academic@yahoo.com>

June 5, 2007

Dear Mr. Raymond Wu:

After a thorough double-blind review, I am very pleased to inform you that your revised manuscript entitled "Component Design, Integration and Service Transformation in Financial Industry" (IJISCM-0628) is scheduled for publication in a forthcoming issue of the International Journal of Information Systems and Change Management (IJISCM),

Best regards,

David

David C. Chou, Professor
Editor-in-Chief, International Journal of Information Systems and Change Management, http://www.inderscience.com/ijiscm/
Department of Computer Information Systems
Eastern Michigan University
Ypsilanti, MI 48197
Tel: 734-487-0054. Fax: 734-487-1941
Email: david.chou@emich.edu

International Journal of Information Systems and Change Management
(ISSN: 1479-3121) 2007 Vol.2 No.2

Component Design, Integration and Service Transformation in Financial Industry

Raymond Wu

IBM Software Group
Email: rwu_academic@yahoo.com

Abstract: *Componentization is the first step in component-based service design; this paper identify component in business, service and physical layers to build up the alignment and inter-layer interoperability. Related research of this paper is service virtualization, which is the final stage in service provisioning. This research paper reviews existing literature and uses examples in the industry to demonstrate how services are derived from business components, and how they are designed and produced in achieving service virtualization. This paper also reviews recent literature in component-based implementation and uses example from the financial industry to review the design process for component-based service virtualization.*

Keywords: component design, enterprise integration, service transformation, financial industry

Biographical notes

Raymond Wu is the IT architect currently working with IBM Software Group, his expertises include, enterprise architecture, integration, metadata, service transformation, financial, telecom and database design. Previously he worked with Oracle, Citibank and Vodafone as enterprise architect.

1 Introduction

Component validation and analysis consists of steps in validation of business components, service components, and physical components; common component parsing analysis; service aggregation parsing analysis; and implementation priority and skeleton. It will be followed by component design. A component conforms to and provides the physical realization of a set of interfaces. First, a component is non-trivial; it is functionally and conceptually larger than a single class or a single line of code. Typically, a component encompasses the structure and behavior of a collaboration of classes. Second, a component is nearly independent of other components. It rarely stands alone. A given component collaborates with other components and in so doing assumes a specific architectural context. Third, a component is a replaceable part of a system.

A component is substitutable for any other component that realizes the same interfaces. This helps during development, as parts of a system can be stubbed, sketched

Copyright © 2007 Inderscience Enterprises Ltd.

and then replaced by mature, robust implementations. It also supports the evolution of a system even after it has been deployed, by making it possible to upgrade and evolve parts of the system independently. Fourth, a component fulfills a clear function. It is logically and physically cohesive and thus denotes a meaningful structural and/or behavioral chunk of a larger system. It is by no means an arbitrary grouping. Fifth, a component exists in the context of a well-defined architecture. From the technical perspective the idea was to eliminate the integration effort by using the "virtual organization" concept. From the organization perspective, this enhances the flexibility of service provisioning and the agility between organization and service; thus, the interoperability across organizations is increased.

The architecture interoperability of this paper is based on component validation, component design, micro process mediation and dimensional services. The realization of this top-down vertical integration approach requires robust mediation to support the inter-communication between component-component and component-repository. In other words, the concept needs a macro framework and physical support in micro integration service bus such as the middleware in realization of run-time services.

The financial industry distinguishes itself from other industries in that time to market is critical, higher security requirements are mandatory, and interbanking service need agility in mediating different parties into a common service infrastructure. For this reason, this paper apply SCA (service component architecture) to deal with component design and interconnectivity in micro mechanism. Common service is a top-down approach which follows the nature of business implementation direction.

A typical financial project consists of domain implementation, architecture and shared services. The domain implementation group is divided into several teams to handle various LOBs (lines of business) as each team will have different areas of business domain knowledge; each team may in fact handle several business lines (i.e., use cases). It's very often the case that the individual LOB team doesn't have adequate concepts and communication to determine the common components, which include common services, common interface and standard data access and data models. This paper would like to review and list these common issues in enterprise integration, along with the proposals for resolution.

i) Too many integration points cause implementation silos
ii) Data fragmentation with duplicate process without aggregation strategy (Curtis, et al, 1999)
iii) Common area is invisible and unable for reuse in a shared model
iv) Lack of priority and focus of strategy slow down business responsiveness
v) Weak enterprise interoperability causes high risk in industry transformation

This scenario is common in the financial industry. In the recent configuration, the conventional credit card service doesn't maintain a robust fraud detection policy at the account level, so many transactions become invalid, the agility and responsiveness of making a proper change is still weak, and many consumers suffer the serious issue that their transaction needs to be rolled back. For this reason, this paper launched the add-on service component of Fraud Policy service to replace the conventional practice and dynamically retrieve cardholder profile, and the Applied Fraud Policy to determine the authorization approval through interbanking. In this manner it is the security profile and preferences of the individual account holder which determine the fraud detection, based on the transaction behavior and customer profile.

2 Literature review

According to L. Yu et al., an obstacle to software reuse is the large number of major modifications that frequently have to be made as a consequence of dependencies within the reused software components. Component-based implementation is the foundation in coping with this bottleneck (Yu, 2005).

The financial industry is one of the most highly complex among the various sectors, and along with the complexity comes high risk. To eliminate the risk, the assumption is: The higher occupancy of common services within the enterprise, the less suffering will be caused in each IT and business transformation, as technology and business innovation and transformation, while creating opportunity, may impact the enterprise seriously if the transformation foundation is not solid. Weakness in enterprise skeleton and in interoperability supported by common components, may cause high risk in each transformation necessary to survival in the next stage. To cope with these issues, this paper followed the enterprise integration strategy in the recent research which includes: validation of business and service components, service analysis and transformation, service design, information aggregation and service provisioning. Our research in enterprise integration in addressing these issues has been divided into four areas: validation, analysis, design and virtualization (Lytras, 2006).

According to P. Kruchten, a component is a non-trivial, nearly independent and replaceable part of a system that fulfills a clear function in the context of a well-defined architecture. Desmond DSouza defines vertical integration in his "Model-Driven Architecture and Integration Opportunities and Challenges," stating that vertical integration comes from different levels of abstraction of the same system, from physical data models, logical data models, networks, applications specifications, component assemblies, business process models and business goals and strategies (DSouza, 2001).

Common interface is also a means to achieve common components. S. Melnik proposes a universal interface which avoids common models and languages; this is the Generic Interoperability Framework which was developed to facilitate the integration of heterogeneous information systems. Within this framework individual communication protocols, data manipulation languages and data are represented in a generic manner while preserving their ontological variety; this process indicates how a mediation infrastructure can be built using canonical wrappers, operational requirements of integration and layered architecture for component design (Melnik, 1999). D. Oberle, S. Staab and A. Eberhart, in their Kaon server design, show several potential sources of semantic metadata. These sources provide input for the framework regarding the semantic management of software components. This includes deployment descriptors, Web and application server configuration files, source code information or annotations and database metadata. The component metadata collector parses this information and automatically integrates it into the ontology as semantic metadata (instances). The inference engine is embedded in the application server and applies the ontology and semantic metadata for reasoning and querying. The application server's core functionality and the administration console both use the reasoning capability. The latter allows the developer to browse and query the ontology at runtime so that configurations may be queried or checked to avoid inconsistent system configurations during development and at runtime (Oberle, et al, 2006).

3 Credit card transaction between cardholder, merchant and interbanking

This paper break down the business component into the following use-case scenarios for further development. The business scenario is related to credit card transaction between cardholder, merchant and interbanking. This paper deconstruct the business components into the following service use cases in sequential order, and use Figure 1 to illustrate the business scenario:

Service use case-1: Buyer sends payment information to the merchant server. Merchant software must send and receive messages, encrypt and decrypt, store public and private keys, and request and receive certificates.

Service use case-2: Merchant software takes payment information from the cardholder and sends it to the acquirer (the merchant's bank). The acquiring institution must receive and authenticate the payment information the merchant received from the cardholder.

Service use case-3: Acquirer sends an authorization request to the issuer over the interbank network. Issuer sends the authorization response to acquirer.

Service use case-4: Acquirer notifies merchant about the status of authorization; if the response is positive, the merchant fills the order.

Service use case-5: Merchant presents the charge to the acquiring bank.

Service use case-6: Acquirer sends a settlement request to the issuer.

Service use case-7: Issuer charges the buyer's credit card account and at regular intervals notifies the buyer of transactions and accumulated charges.

Service use case-8: Buyer pays the charges to the bank. Acquirer credits the merchant's account.

Interoperability of service provisioning and virtualization have become focus in recent research. In the conventional credit card service, the interbanking doesn't maintain a "fraud detection policy" service at the account level. Mostly this shortage is the result of conventional silo implementation and limitation of multi-site interconnectivity such that a daily program is used to update the hard-coded matrix within the whole authorization process.

The issue is with a non-component-based environment in which implementation can cause high impact on the enterprise, and may even be unachievable between different organizations having a common metadata to handle. These are shown in Figure 2, which illustrates the add-on component provisioning a new service in fraud detection as a dynamic option based on customer profile. Our newly launched add-on services include a component-based Fraud Policy service which, as explained in the Introduction, will replace the conventional practice and dynamically retrieve cardholder profile and the applied fraud policy to determine the authorization approval through interbanking, so each account holder's security profile and preference for the card issuer or dealers will determine the individual fraud detection based on the transaction behavior and customer profile. This is the service transformation and provisioning example which mediates the enterprise components in forming the skeleton. This skeleton requires aggregation and composite services, and semantic interoperability as well.

The semantic interpretation from business to service, and from service to technical, needs a deconstruction process to work on those selected business components, so that each business component can be deconstructed into business use cases for further

breakdown. Service components derived from the business use case are the mediation between business and technical elements which integrate all business semantics, interfaces and technical configurations into a logical component. Initially all the service components are at an atomic level (Staab, et al, 2001) (Valkeapää, et al, 2006).

Service normalization means grouping all atomic service elements from fine-grained to coarse-grained. The conventional credit card service doesn't maintain a fraud detection policy service at the account level; it is maintained as a generic rule applied to all cardholders within the entire authorization process.

Prior to the component design, the business component selection, deconstruction for service component validation and analysis are required in component-based design for enterprise integration objective. This paper will not detail the process as these areas were the focus of our previous research. This paper introduced a group decision model in the validation process, so business components with their associated service components are selected and validated through the decision process. This paper further detailed the rational model in analyzing common components and aggregated services by using a use case parser to determine enterprise components in forming the skeleton of integration. Aggregation is a kind of service normalization in grouping all atomic service elements from fine-grained to coarse-grained components. This paper also introduced the component design and interconnectivity for how the components communicate with each other and how the micro process of the business semantic is derived in each component. Figure 3 further shows the referential integrity of component interconnectivity for how the components relate to each other.

4 Physical component design and aggregation

Figure 4 shows the design of components and interconnectivity. The component element is the physical element that mediates service and information. The design of service components follows our previous strategy, by using business semantic injection to take the context into the semantic and micro processes, so that each component has its contract-level responsibility in provisioning a subset of services, and the interface helps the coordination between components and metadata build the foundation of interoperability within the service layer and between layers. In the following paragraph, this paper take the results from component design into the service level, and meanwhile this paper apply automata theory to build up the rule-based service aggregation at the enterprise level. As this research is still at a pioneer stage in applying automata theory in service provisioning and aggregation, this cannot be accomplished within a short period, and further research efforts will be undertaken following the current work (Barcia, et al, 2005).

The string $\Phi(r_n, r_m) = \Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m$ is a path from r_n to r_m called "service route" and $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$ a "composite service" between r_{i-1} and r_{i+1}, $\Phi_{i, i+1}$ is used to denote composite service. When frequency $F(\Phi_{i, i+1}) = F(\Phi_i \times \Phi_{i+1})$ is higher than threshold σ, the concatenated alphabet (part of the string) will be aggregated from $\Phi_i \times \Phi_{i+1}$ into $\Phi_{i, i+1}$, this implies the original state r_i between r_{i-1} and r_{i+1} will no longer exist for this composite service.

5 Component-based information metadata interoperability

From the information metadata perspective, to achieve the service provisioning in single view of the customer, all fragmented data sources should be aggregated into metadata and business intelligence to build the service virtualization foundation based on the

integrity between business, service and information. M. Schmalz argued that database schema and application code can be efficiently derived from various types of schema representations, particularly the relational model, and supports comparison of a wide variety of schema and code constructs (Schmalz, 2003). Once the customer's preference and transaction behavior is visible, high-value services can be added on top of the information metadata, which represent the accuracy of customer, summary information and business intelligence; this is the information aggregation metadata concept.

The interoperability skeleton is an example of how the micro interoperability in the service layer, aligned with the component layer in micro mechanism, provisions the macro service in business. The foundation of this skeleton is interconnectivity between service components by using common components and aggregated components. This paper further develop the service virtualization on top of this interoperability skeleton, which is called enterprise vertical integration, in extending the integration across industry and geographical areas. This is the virtualization service in a horizontal direction; once the industries of different regions build up their semantic interoperability in three metadata repositories (business, service and information), they can synchronize their business, operation and strategy, so that the limitations of business, geographical area, products, application and data structure become transparent.

6 Service transformation

The transformation therefore requires an event to incur the service with the intelligence to either predict the next service or to verify that the called service is recognized. State concept discussed above is used to mediate the business, service event and information in the "time" dimension. The state of the service layer and the business-service-information coherence are changed once an event is triggered and the service is successfully executed. The alphabet read in automata is leveraged to service component trigger and execution while the state change is leveraged to transformation within the service layer. At the very beginning of a use case, the first event triggers service Φ_1

$\varepsilon^1 : r_0 \times \Phi_1 \rightarrow r_1$

When the service event Φi ($i <> 1$) is read and triggers the service component, the state transformation can be expressed as follows:

$\varepsilon^i : r_{i-1} \times \Phi_i \rightarrow r_i$ with transformation as follows:

$\varepsilon^{i+1} = (r_i) \times \Phi_{i+1} \rightarrow r_{i+1} = (r_{i-1} \times \Phi_i) \times \Phi_{i+1} \rightarrow r_{i+1}$

which is in equilibrium to $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$, and derives

$\varepsilon^{nm} : r_n \times (\Phi_{n+1} \times \Phi_{n+2} \ldots\ldots \times \Phi_m) \rightarrow r_{m\ (n<m)}$. This is called "semantic event aggregator."

As the final step of a use case, the final event trigger service Φ_n

$\varepsilon^n : r_{n-1} \times \Phi_n \rightarrow r_n$, Φ_n is a terminator, an example of which is the settlement of a daily transaction.

In any component context a formal interface contract for the invocation syntax is important. The semantics issue (the meaning of parameters and so forth) has to be solved as well (domain modeling). This is crucial to any business-to-business (B2B) and dynamic invocation scenario. Such scenarios are cornerstones of the service component vision of being flexible and agile in response to the new business needs in a world of mergers and acquisitions, business transformation, globalization and other relevant factors (Zimmermann, et al, 2004).

A tool called JFLAP guides the user through several proofs of theorems that focus on transformations of languages from one form to another in which the forms are automata and grammars. JFLAP aids in automata visibility; users are given a visual representation they can interact with in addition to the formal textual representation. This alternative hands-on representation provides a means for an interactive automata theory (Gramond, et al, 1999).

The string $\Phi(r_n, r_m) = \Phi_{n+1} \times \Phi_{n+2} \ldots \times \Phi_m$ is a path from r_n to r_m, called "service route" and $r_{i-1} \times (\Phi_i \times \Phi_{i+1}) \rightarrow r_{i+1}$ a "composite service" between r_{i-1} and r_{i+1}, and symbol $\Phi_{i, i+1}$ to denote a composite service.

When frequency $F(\Phi_{i, i+1}) = F(\Phi_i \times \Phi_{i+1})$ is higher than threshold σ, the concatenated alphabet will be aggregated $\Phi_i \times \Phi_{i+1}$ into $\Phi_{i, i+1}$, this implies that the original state r_i between r_{i-1} and r_{i+1} will no longer exist for this composite service.

When the business architect checks the result from semantic event aggregator - ε^m which is more an upper-layer use case scenario, service route - $\Phi_{i, i+1}$ which is in the service layer and its frequency - $F(\Phi_{i, i+1})$, if $F(\Phi_{i, i+1})$ is much higher than threshold σ, then from the business perspective the event aggregator ε^m either consists of many composite services or is business critical. This means the aggregation service is the key within business and is crucial in daily operation from service perspectives, and that fine tuning is feasible from a multi-fine-grained boundary to a coarse-grained service. This is called "main route" of the enterprise skeleton; in other words this is the highway between A and B. Virtual coarse-grained service eliminates the boundary and leverages many fine-grained components which can be easily configured by adding or modifying the component dynamically to achieve the broader coverage of main route services. For this reason main route is one of the crucial services in enterprise integration. One example is in the telecommunications industry that incorporates an open service account, an open billing account and provisioning options for a new customer.

This constitutes a main route for "new customer service" coarse-grained service, though it is aligned with fine-grained components to support customer preference, service criteria and billing options which are branches within this main route. The importance of determining the common component and service aggregation is in building up an enterprise skeleton of vertical interoperability between business, service and information. This task is crucial, as in the business domain analysis stage, the impact area and scope to business components are minimized in decision making. Additionally, the business component is deconstructed into service components and physical elements for top-down alignment while the common component is not simply a symbolic service but instead requires a bottom-up approach to work on common areas; this serves to fine tune the common components and interfaces from the perspective of enterprise integrity and performance. Figure 5 shows how common components and composite component service aggregation form the skeleton of enterprise integration; the skeleton extends its service to the derived service and forms the interoperability with upward linkage to business and downward to physical components. The service aggregation is also important as the top-down deconstruction followed by bottom-up composition forms the skeleton of vertical interoperability and potential component-based implementation of the service process.

7 Service virtualization

Once the vertical skeleton and service extension are built, the interoperability reflected from the business layer has the capability of integrating cross-organization and cross-geographical constraints, as fully transparent services become possible if each system or

organization follows an industry standard by using metadata strategy and an industry framework. There are three steps to reaching service virtualization (Hunter, 2001):

1. Enterprise consolidation, to identify the unique source of truth within an enterprise, and use its metadata within business, service and information as a corporate standard, so when other systems and applications are consolidated into the enterprise repository, there is a standard and guideline to follow.

2. Vertical enterprise integration, as when other systems are consolidated into the enterprise, there will be some metadata out of scope and the need for further integration efforts to streamline the vertical integration.

3. The final step is the service virtualization which is the objective of our enterprise integration; in this step, it is required to make sure our repository is holding enough and accurate information within the industry standard to integrate the enterprise to the external, so our metadata in business, service and information are provisioning the services through industry and geographical locations in a transparent environment and in a horizontal direction.

8 Component design and integration in the financial industry

The micro process in component design follows the top-down direction and takes the deconstruction output to derive the service component. This paper use the interbanking example of the credit card industry to demonstrate the transaction between cardholder (payer), merchant (payee), bank (transaction handler) and credit card authority (interbanking). This paper also deconstruct the business component of credit card transaction into several use cases, as follows. 1) Consumer contacts merchant to encrypt and decrypt security keys, and to request certificates. 2) Payment is transferred from merchant to acquirer (the merchant's bank), and then to issuer through interbank network to authenticate the payment information. 3) Acquirer notifies merchant about the status for merchant to fill the order. 4) Merchant requests acquiring bank regarding settlement. 5) Issuer charges the consumer's credit card and notifies cardholder. 6) Consumer pays fee, and acquirer then credits merchant for the exact fee.

The conventional design doesn't have the dynamic control to have customer insight while the issues are resolved in component-based design. This is the service transformation and provisioning example which mediates the enterprise components in forming the skeleton. Service components derived from the business use case are the mediation between business and technical elements which integrate all business semantics, interfaces and technical configurations into a logical component; initially all the service components are at the atomic level (Staab, et al, 2001) (Valkeapää, et al, 2006). In our component validation and analysis research, this paper applied a group decision model in the validation process, so the business component with its associated service components are selected and validated through the decision process. This paper further detailed the rational model in analyzing common components and aggregated services by using a use case parser to determine enterprise components in forming the skeleton of integration. Aggregation is a kind of service normalization in grouping all atomic service elements from fine-grained to coarse-grained components.

This paper also introduced the component design and interconnectivity for how the components communicate with each other and how the micro process of business semantics is derived in each component. The component element is the physical element that mediates service and information; the design of service components follows our previous strategy, by using business semantic injection to take the context into the semantic and micro processes, so that each component has its contract level

responsibility in provisioning a subset of services, and the interface helps the coordination between components and metadata build the foundation of interoperability within the service layer and between layers.

9 Conclusion

An example of a typical issue in a credit card scenario is given for how to narrow down the root issue within a small scope as a service component while maintaining its alignment to both business and technical components; this service component architecture speeds up the change and dramatically enhances the integrity and responsiveness. An example of service provisioning would be the case where several key industry vendors have built up an industry framework for service providers in that particular industry to follow, starting from business domain analysis and working through to component deployment. So each service provider benefits from the industry common services and common interface, particularly those B2B, B2C, merger and acquire cases. Any business activity becomes easier when the various parties involved have a common understanding and consensus on business terms, service definitions, interfaces and component specification, and these become transparent and common practices.

The enterprise integration life cycle consists of validation, analysis, design and virtualization. Enterprise interoperability is built on the foundation of component-based design; this paper applied both macro and micro process in constructing the foundation, so the components is aggregated into common components and aggregation services. This encompasses component-based design and interconnectivity, transformation of service aggregation through rule-based design, service provisioning in a particular dimension and service virtualization. In component design this paper introduced the micro mechanisms within components as well as component interconnectivity. The important concept derived in this research is that once the metadata of business, service and information constructs an industry standard, the organization's vertical skeleton has the opportunity to integrate across multiple organizations for an easier global practice; this is a long-term industry objective which called service virtualization.

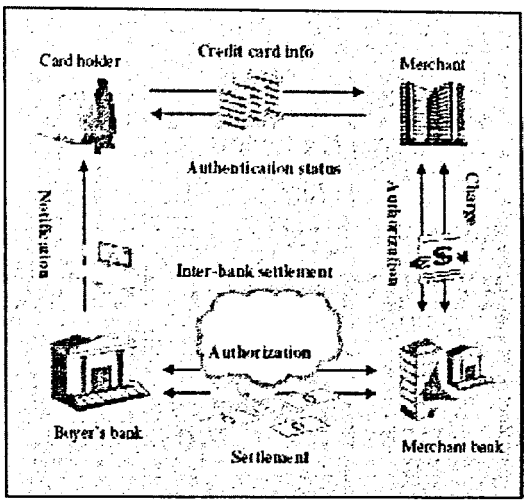


Figure 1. Business use case in the financial industry

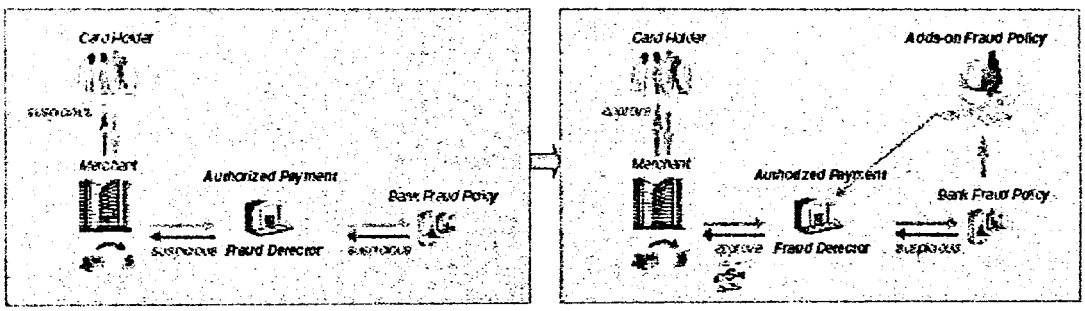


Figure 2. Loosely coupled component add-ons solution for fraud detection

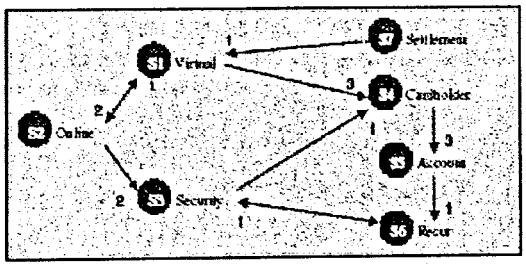


Figure 3. Referential integrity of component interconnectivity

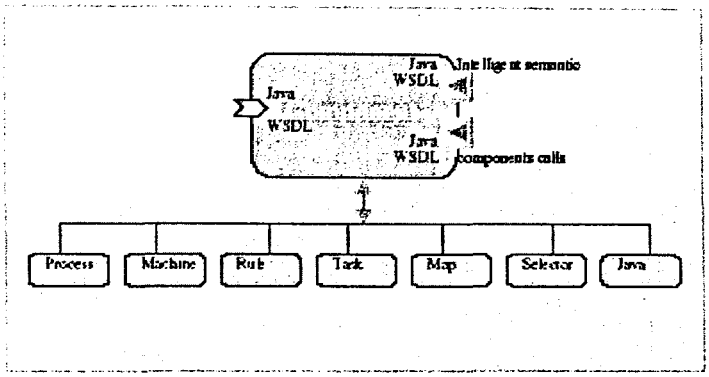


Figure 4. Component design in provisioning the service layer

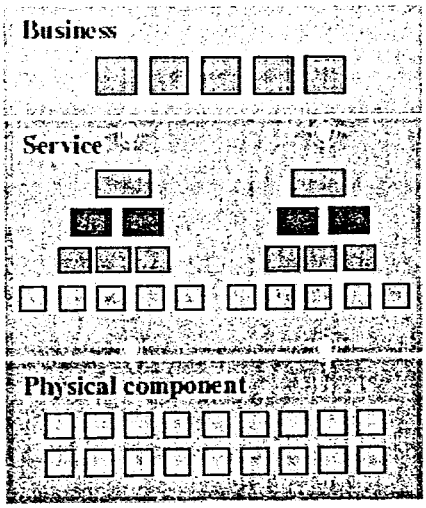


Figure 5. Transformation of common components and service aggregation

References

Curtis, K., Foster, P. W., Stentiford, F. (1999) Metadata - The Key to Content Management Services, Proceedings of the Meta-Data '99, The third IEEE Meta-Data conference, April 6-7, 1999, Bethesa, Maryland, USA

Yu, L. (2005) Categorization of Common Coupling in Kernel-Based Software, 43rd ACM Southeast Conference, 2005

Lytras, M. (2006) The Semantic Electronic Government: knowledge management for citizen relationship and new assessment scenarios, Electronic Government, an International Journal 2006 - Vol. 3, No.1 pp. 5 – 17

Dsouza, D. (2001) Model-Driven Architecture and Integration Opportunities and Challenges, Kinetium 2001

Melnik, S. (1999) Generic Interoperability Framework, Distributed and Parallel Databases Journal 1999

Oberle, D., Staab, S., and Eberhart, A. (2006) Semantic Management of Distributed Web Applications, IEEE Distributed Systems Online, vol. 7, no. 4, 2006, art. no. 0605-o5001

Staab, M., Stojanovic, N., Studer, R., and Sure, Y. (2001) Semantic Portal - The SEAL approach, to appear in: Creating the Semantic Web., MIT Press

Valkeapää, O., and Hyvönen, E. (2006) Semantic Annotation with Browser-based Annotation Tool, 1st Asian Semantic Web Conference

Barcia, R., and Brent, J. (2005) Building SOA solutions with the Service Component Architecture, IBM Research

Schmalz, M. (2003) EITH – A Unifying Representation for Database Schema and Application Code in Enterprise Knowledge Extraction , 22nd International Conference on Conceptual Modeling

Zimmermann, O., Krogdahl, P., and Gee, C. (2004) Elements of Service-Oriented Analysis and Design, IBM Academy

Gramond, E., and Rodger, S. (1999) Using JFLAP to Interact with Theorems in Automata Theory, Thirtieth SIGCSE Technical Symposium on Computer Science Education, ACM Press 1999 page 336-340.

Hunter, J. (2001) MetaNet - A Metadata Term Thesaurus to Enable Semantic Interoperability Between Metadata Domains, Journal of Digital Information, Volume 1 Issue 8

A7

Service Virtualization Analysis and Review for investment banking

IBM

Version 1.0
October, 2006

Version Number:	1.0
Date:	10/12/2006
Author:	Raymond Wu, Consulting IT Architect
	IBM Software Group

For the presentation to ASEAN enterprise architecture

Summary

Evolving business needs and new technologies are spurring, Today's clients are comfortable receiving a large part of their investment information from one source, but not all their investment choices. In order for banks to successfully attract and retain clients they must continue to pursue an open architecture investment strategy.

Services Virtualization in Investment Banking

A common service and information sharing is required, so those duplicate data retrieval and processes can be eliminated. Once work load is minimized, the integrity and performance will be improved and IT cost can be much lower from lighter computing load

Background

Services firms to change the ways they use IT to cut costs

1. Componentization is a must, a lack of industry process, message and semantic standards has hindered the move to component based architecture
2. Front/ back-office components interconnectivities
3. A transition from a control-based industry culture to a collaborative industry culture is required
4. Streamline operations and improve agility

What need to achieve for high ROI ?

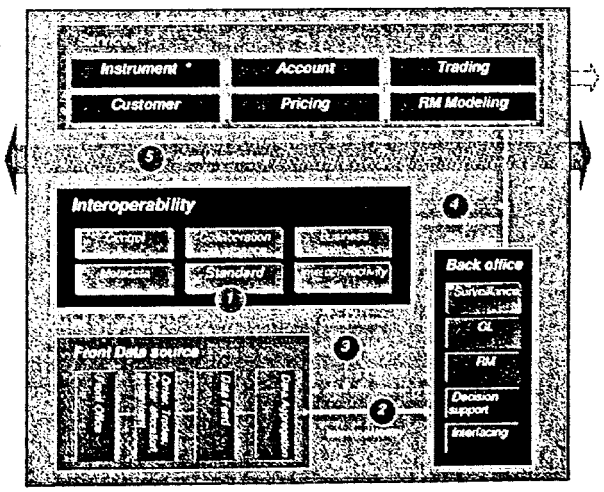
- Through focused business areas and customers
- Business Intelligence in investment analysis
- Dynamic advisory in investment decision
- Responsive Business Model
- Faster time to market
- Enterprise B2B and B2C integration

What cause high impact in IT cost ?

- Highly heterogeneous applications
- Information fragmentation
- Weakness in Instrument provisioning
- Weakness in linkage between front and back offices

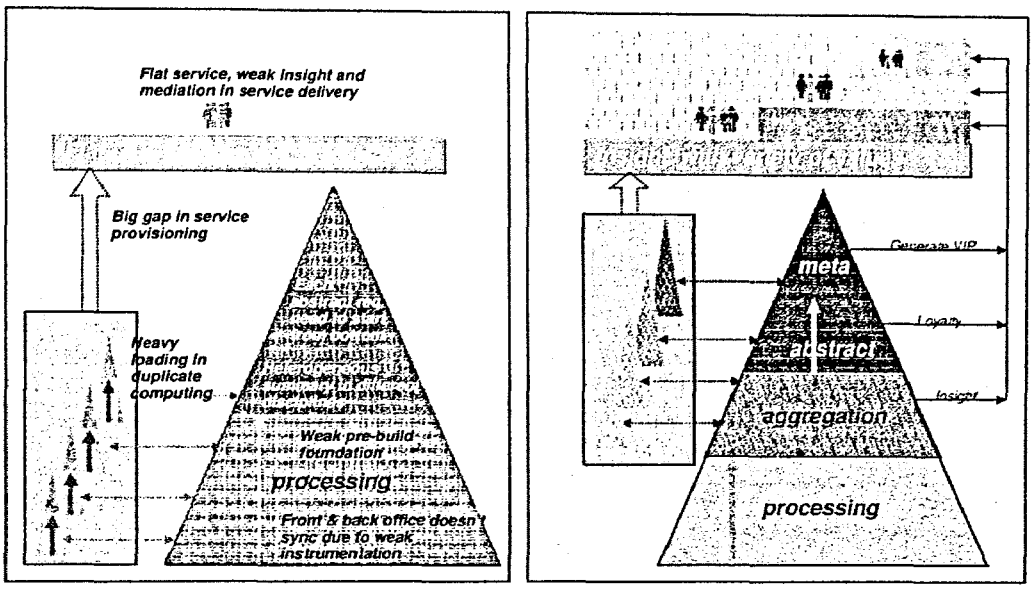
Open architecture allow brokers and institutional clients to perform their trade activities like purchases, liquidations, exchanges, and cancellations over the web, to support its renewed thrust on e-services and to provide an end-to-end solution, ranging from basic account opening and querying to placement of orders, redemptions and executing other transactions straight through. Instrument usually cover;

- any financial security such as a bond, stock, check, etc.
- Money market securities (such as Treasury Bills, government bonds, or commercial paper)
- capital market securities (such as mortgages, certificates of deposit, or long-term bonds)

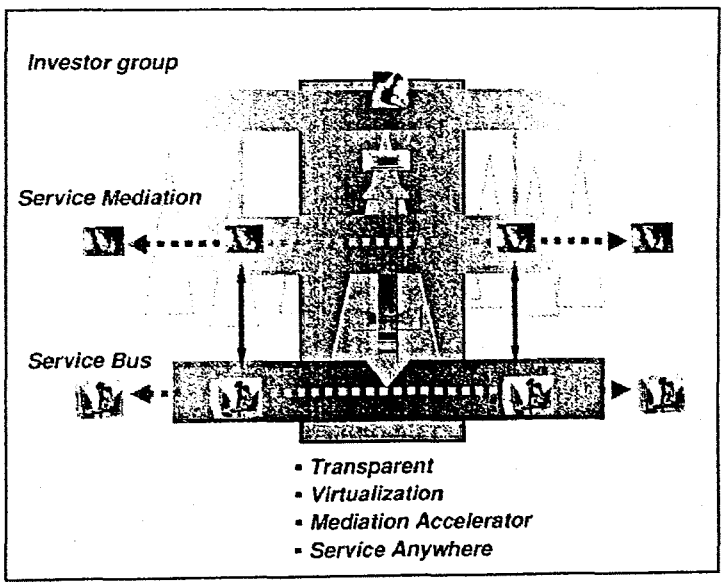


Computing economy in Meta Data Aggregation

- IT Increase ROI by Precise Analysis, Decision support and Customer Insight
- Cut cost by pre-building data, eliminating duplicate process and mass data retrieval



Open architecture allow brokers and institutional clients to perform their trade activities like purchases, liquidations, exchanges, and cancellations over the web, to support its renewed thrust on e-services and to provide an end-to-end solution, ranging from basic account opening and querying to placement of orders, redemptions and executing other transactions straight through.



IT Strategic Assessment

Realization – Service Virtualization

- Maturity Assessment
- Meta data Mapping and Broker Mediation
- Integration and Geographical Transparency
- Financial model, Business Process & Service Component
- From Service Bus, Common service to Visualization

IBM WebSphere Studio Application Developer development tool allows Application Developer to generate JavaBean that makes JDBC calls to stored procedures running in DB2 or other database.

1. The Investment Banking Group's IT policy need to decide business logic should reside in stored procedures or at the middle tier (application server) level.
2. Decide whether Enterprise JavaBeans (EJB) components is needed. if stored procedures is selected, EJBs would only offer transactional behavior that needlessly duplicated features those stored procedures already provided.
3. Decide whether WSAD/IE is needed to automatically generate the WSDL document and Web service deployment artifacts necessary to deploy JavaBean into the WebSphere Web services runtime environment.

Take a usual case as example, a Microsoft Word template uses an account number to invoke a macro that queries an application hosted on a back-end WebSphere Application Server. To build a bridge between the MS-based client and J2EE-based server environments, both communicate via a SOAP-based Web service. The client macro performs a SOAP RPC call to a Web service running on WebSphere. This Web service, which is simply a facade for a Java class that accesses a JDBC data source, in turn triggers stored procedures in DB2 (or other database). For e-business of an actual business scenario in the financial services industry requiring application integration, Web services can help bridge gaps between organizations within an enterprise with very different IT philosophies. IBM Web services produce a collection of general best practices for Web services architecture and design.

Business Component Decomposition

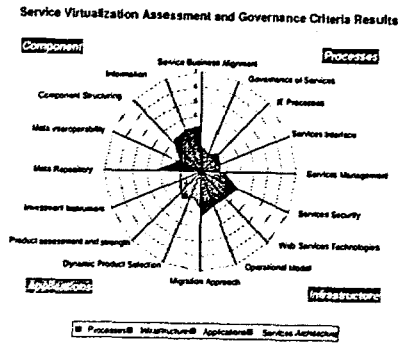
Modeling and identify the change
Understand Business Goals to determine related Business Processes
Filter & Focus work on highest priority parts of the business model the Business Processes traced from the business Goals Accelerate using Business Processes Template to analyze and Improve the Business Process Design.

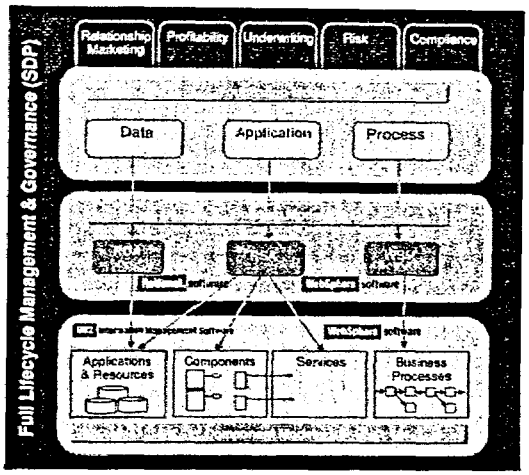
Decide and validate the change
Request Change to Business Process & Policy
Enter Change Request traced to Business Requirement Review Change Request.

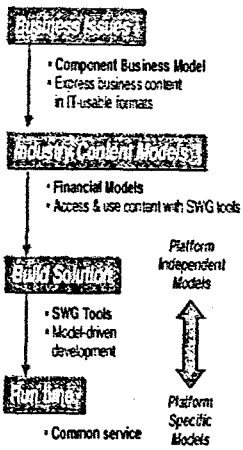
Govern and Make Project Portfolio trade-off decisions
Review Business Goals, Analysis
Requirements traced to the Change Request
Understand the Business Drivers and justification motivating the request

Plan and execution the change - Approve/Deny Change Request
Governance – eSignature for Approval
View Approved Project/Change Requests
Governed, Auditable Automated Development workflow
Evaluate Project Portfolio to make tradeoff decisions
Create & Review Project Resource Requirements
Create Iterative Project Plan/WBS
Staffing the Project
Search For, Identify and Assign Project Resources

Architecting the UseCase and Modeling the business analysis
Analyze proposed change to business process from a Software Architectural (UML) perspective
Establish/Visualize traceability of Requirements to Bus Model
Search for and Assess Re-Usable Assets
Cost-effectiveness, Accelerated development, Consistency
Design Architecture and Services
Determine the Necessary Service Collaborations
Transform architecture to executable assets
Execute against common process and project plan
Process Guidance as needed
Make all changes within governed infrastructure
Transform Design to Code
Accelerate results with transformations
Review and finish coding transformed design model (services)
Code business logic within the transformed design constructs
Test assets
Deploy New Service to Integration Test Environment
Governed, Auditable Project Workflow
Ensure Quality
Test Plans, Cases and Automated Tests traced to Requirements
Test to ensure Requirement(s) are fulfilled
Governed, Auditable Project
Component Driven by CBM, Financial Models, MDD, Rational software







- The key elements of this meta-model encompass the Service Consumer view and the Service Provider View
- The Service Consumer adopts a business-aligned, business driven perspective, looking for specific kind of functionality provided by a Provider with a certain Quality of Service. This includes the Services View required to fulfill the Business View.
- The Service Provider in turn understands the functionality and QoS of the domains it supports and realizes the services through managed IT components that are governed at an Enterprise scale and are considered to be enterprise assets for the provider organization.
- Each may have their own business views that converge to deliver functionality and quality of service required and provided, respectively.
- The Method View depicts the concepts necessary to the domain of building SOA through service-oriented modeling, analysis and design
- The Fabric View is also hidden, but describes dependencies on the infrastructure and fabric that provides the underlying support, such as the Enterprise Service Bus.

A8

Oracle Strategy in Financial System Architecture and Integration

金融服務管理系統解決方案 及最優化戰略

Part I - Architecture

For the presentation to Oracle Financial Alliance in Greater China

ORACLE
甲骨文
2005年1月3日

Represented and presented by:
Raymond Wu
Principle Consulting Architect

1. 整體技術針對金融服務業的解決方案

全球化、簡單化、標準化、創新 以及自動化，這些原則確保應用軟體、資料庫、應用伺服器、協同作業、開發工具) 與服務在競爭激烈的環境中奠定邁向成功之路。 以下是 Visa International 的資深副總裁 Gretchen McCoy 提出個人看法 - 當你開始使用這一功能強大的軟體時，不僅能夠解決過去存在的問題，而且還能夠利用 Oracle 的完整平臺將你的整個企業轉變爲電子商務企業。一旦擁有了一個整合的系統，你就會認識到它是多麼重要。"

Oracle 利用其領導能力、全球合作夥伴關係、行業遠見、知識及經驗來提供專門爲金融行業開發的解決方案，來滿足銀行金融業的需要。借助 Oracle 提供的金融服務業解決方案，用戶將能夠在競爭激烈的環境中，致力於:

- 構建一個更高效的機構
- 滿足嚴格的法規要求
- 提高從客戶的盈利能力評估
- 管理和提高按風險調節的資本回報率
- 滿足新巴塞爾協定的規章要求
- 有效地管理資產負債表風險
- 制定企業發展和績效最優化戰略
- 實現企業希望的業績

1-1 系統軟硬體

下表所列是 Oracle 針對一般金融服務產業提供的解決方案：

金融服務解決方案				
整合集成	服務	金融分析、計劃與報告	市場與銷售	管理
✓ 分散式伺服器 ✓ Web 服務 ✓ 安全性 ✓ 整合集成與工具組 ✓ 門戶管理 個性化管理 無線管理 遵從 J2EE	✓ 電話銷售 ✓ 高級入站 ✓ 交互中心 ✓ 電子郵件中心 ✓ 線上支援	✓ 總帳 ✓ 財務分析器 ✓ 績效分析器 ✓ 風險管理器 ✓ 平衡計分卡 ✓ 商務智慧 ✓ 資料倉儲 ✓ 全球整合系統 ✓ 金融資料管理器 ✓ 資金轉讓定價 ✓ 預算與規劃 ✓ ADI ✓ 活動管理	✓ 市場營銷 ✓ 線上客戶管理 ✓ 線上銷售管理 ✓ 銷售分析器促銷 ✓ 線上資金轉讓定價	✓ 採購尋源 ✓ 線上採購 ✓ 購買許可 ✓ 固定資產管理器 ✓ 人力資源 ✓ 薪資管理 ✓ 福利管理 ✓ 數位教學 ✓ 專案管理

紅色代表必備項目　　綠色代表考慮項目

1-2 滿足系統功能基本需求

中心控管總帳 (Single View of Account)

中心控管總帳架構保障資料正確性，透明性，稽核性，及嚴密控管. 它同時維護單一總帳內部本國幣及外幣以及所有金融平衡的控管 其要項;

- 實際平衡
- 預算平衡
- 概括帳戶平衡
- 統計性項目
- 平均性平衡

客制化與控制 (Customerization Flexibility)

- 使用者定義的產品分類：報表需求不同，定制的帳戶圖表就包括多種情況（例如償債能力、利率敏感度）
- 使用者制定的集聚 (consolidation level)資料：在使用者控制的級別上，集聚獨立處理過程的資料
- 使用者自定的處理規則：根據產品或金融產品特徵，選擇資產負債表財務會計科目進行處理，使用者自定的期間 (Term)：爲進行收益敏感度分析，按月或按年混合模型期間；按天使用償債能力報表；設置獨立類比期間的缺口期間

擴展架構 (Scalability)

對於企業級風險管理，解決方案的可伸縮性是至關重要的，它必須提供高處理效率、健壯性和分析的可擴展能力。系統應可運行在高度優化、多進程、32 位元框架之上，可以最大限度地發揮硬體平臺的性能。

應採用客戶/伺服器方式的硬體架構，及選擇伺服器處理能力的等級。當需要增加則可擴展至更大能力、多高速度，與需求相匹配的硬體。

建置於開放式系統平台上，並透過共同之介面標準，整合現有各業務電腦系統建置具備多用途之共用平台資料倉儲。開放式系統完全支援讓其他 SQL 相容軟體工具以自由形式 (ODBC compliance)存取。資料庫和資料模型可增加欄位。而在建置報表時，系統提供使

用者可選擇增加欄位。

完整及全面性的維護資料 (Data Integrity & Maintainability)
- 金融業全行資料模型包含詳細資料（交易及客戶帳戶細目皆有）與總覽金融資料。
- 細目資料可集中供任何模型所需，所有金融決策支援系統亦可使用相同之資料。

快速及彈性地存取資料 (High Speed Accessibility)
- 金融業全行資料模型採用開放架構之關聯式資料模型。
- 允許建立自訂資料欄位與表格，任何資料欄位選取使用者自訂值。
- 可讓任何採用遵循 ODBC 標準之 SQL 語言之應用程式存取使用。

加總 (Aggregation)
- 評估歷程資料內是否不再需要之詳細資料，如帳號、交易量等；
- 利用將相同現金流特性（如起始日期、付款週期等）之細目資料群組集中，強化金融決策系統產生之現金流；
- 可供多重使用者存取使用之預先建立報表加總；
- 藉由集中客戶資訊及收益資訊，評估特定商品（存款、抵押等）之客戶收益，並可進行資訊分析與模型化。

平衡與控制 (Balancing and Flexibility)
資料完整性與強化資料之控制為金融效率管理之關鍵。要讓資料有價值，必須讓資料有效，資料品質必須得到控制，還有必需能夠調整。否則，有可能根據衝突之假設做出決策。
- 資料驗證- 驗證資料與控制品質 (日期)
  資料調整與校正工具 - 當發現資料不一致之情況時，允許即時更正，調整一般帳務加總與標準控制號碼之細目資料，提供重要的反饋更正相關系統中之錯誤資料，其設計用於編輯與管理帳戶階層與不同集合階層之資料
- 確保資料完整與正確供模型化使用 - 金融決策系統之資料驗證與編輯能力確保全部資料皆有意義，不論絕對或相對於其他資料欄位，及符合全部模型化需求。
- 加總「部分」資料求得「全部」財務狀況 - 關鍵調整及控制文件皆由檔案格式或人工輸入擷取（例如一般帳務資料）。然後這些文件將與細目資料加總結果進行比對，確保細目資料之完整性與無重疊情況發生。
- 細目資料易於整合模型化 -帳目設計之多重圖表能指定給每個細目帳戶資料錄促進模型化。
- 允許獨特化 - 金融決策系統環境的特殊之處在於完全的客製化符合銀行與其商品混合之獨特需求。
- 自單一資料庫之資料建立不同的檢視 - 金融決策系統之「層級」指定程序預先定義了多種利用資料錄之方式，依照商品類型之商品類型、依照分行之分行、依照分行之商品類型、依照客戶分類之分行等。一旦「層級」定義完成，即可在「樹狀」階層結構裡使用，提供完整的分析能力。
- 需要時回存原始資料 - 編輯程序之每個稽核步驟皆儲存以確保資料的每個更新皆可追蹤到。
- 提供完整的驗證檢查- 檢查資料格式、內容值及資料項目間之關連性一當資料載入時一以確保資料有效性。
- 自動化的重大事項調整功能 -容易自調整與控制文件擷取資料讓自動化調整成為可能。
- 提供彈性化的編輯 -編輯程序可藉由鍵入或分析程序自動輸入，而稽核步驟將自動產生。
- 促進編輯程序 -根據資料驗證及調整程序之結果，有多樣的編輯選擇提示使用者。公用程式藉由分析程序指導使用者資料錯誤校正或找出遺失資料之步驟。
- 提供更正後之反饋資訊給次級系統 - 製作細目報表或電子檔案標明原始資料錯誤之處。而後允許在您工作之處或次級系統中進行修正之動作。

1-3 系統需求範例

- Oracle 系統使用者端採用最佳化的執行方式，一般使用者，包括公司內部和外部使用者並不需要 high-end 的機器. 一般而言，程式開發人員使用 PC 至少 512MB RAM，使用者使用至少 256 MB RAM，外部使用連線頻寬至少 1MB.
- 應用程式伺服器 (AP server) : 2*CPU , Ram : 4G , HDD:36G *2-o.s mirror ; 73G*3—AP Raid

5,Tape drive: 20/40

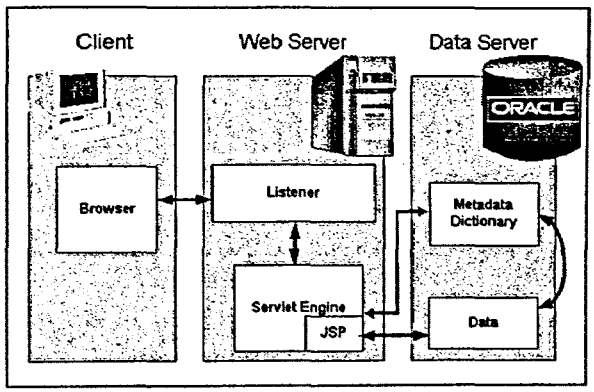
- 資料庫 **(DB server)**: 2*CPU, Ram : 4G , HDD:36G *2-o.s mirror ; 73G*3—DB Raid 5,Tape drive: 20/40
- **Client** 軟硬體需求
  IE 5.5+ or netscape 4.6+ General PC with 256MB RAM
- **Oracle Applications**
  Oracle E-Business Suite 是完整的一套商業應用軟體，適用企業流程的管理與自動化。
- **Oracle Application Server**
  Oracle Application Server 率先整合開發和配置電子商業入口網站、交易應用軟體與 Web 服務所需。
- **Oracle Collaboration Suite**
  Oracle Collaboration Suite 是單一整合式系統，適用於貴公司所有的通訊與資料，包括 web 會議、檔案、電子郵件、語音郵件、傳真與無線。
- **Oracle Development Tools**
  Oracle Developer Suite 是完善整合的環境，結合應用軟體開發與商業智慧工具。
- **Oracle Database**
  Oracle 提供高可用性 (availability), 可靠性 (reliability), 安全性 (safety) 及延伸性 (scalability) Oracle Database 10g 是業界第一個專爲網格運算而設計的產品，藉由將伺服器叢集在一起而形成強大中央處理器，進而降低 IT 資源成本。
- **Oracle Entrise Manager**
  Oracle Entrise Manager 是單一的整合式產品 適用於監控和管理不同資訊技術環境中的 Oracle 軟體基層架構以及應用軟體與專業服務。

2. 技術架構

2-1 Server 與 Client 端操作平台

- Database: Oracle 9i
- Server : Oracle GL can run on any major platform, including UNIX, linux, WinXP etc.
- Client : IE 5.5+ or Netscape plus Java Plugin Jinitiator
- System Architecture : Oracle 建議採行 3-tier system architecture.

Oracle Form uses development suite to customize UI and OA Framework uses html to customize UI.



使用者平台：瀏覽器規格及安裝軟體需求
IE 5.5+ or netscape 4.6 + Jinitiator 1.1.8.16
硬體結構爲 Browser/Server 和 Client/Server 的混合模式；對於前端 GL 業務使用者和系統管理員、報表管理員使用者需要相應的 Client PC 機，爲 Client/Server 結構運行環境。對於大量的資訊使用使用者只需瀏覽器接入 Intranet 即可，爲 Browser/Server 結構，如需要發佈資訊到 Internet，則需要設定防火牆等安全機制。

系統後端需要一台資料庫伺服器(Database Server)和至少一台應用伺服器(Application Server)：
資料庫伺服器安裝 Oracle9i 資料庫和 GL 資料模型的資料結構，及 GL Server 端軟體。
應用伺服器安裝 Oracle9iAS 應用伺服器，包括運行 Discoverer, Oracle Form 的 Server 端 件。
資料庫伺服器要求是 Unix 硬體平臺，GL 支援 Unix 硬體，包括 IBM、HP、Sun 的機器，對硬體型號沒有具體要求，應用伺服器建議是 Unix 平臺（也支援 Win 2000 Server），型號不限（IBM、HP、Sun），硬體設定 CPU 至少 2 個（主頻 >1GHz），記憶體 >2GB，硬碟 >50GB。

軟體資訊產業，經過多年的產業變革，逐漸有了愈來愈清晰的輪廓。西元80年代，引領風潮的最佳

元件(Best of Breed)之整合方式，在歷經了90年代殘酷的競爭後，使用者開始發現，將完美的烏托邦世界架構在如此競爭的世代中，期待各個軟體廠商長期合作是不切實際的。各家合作廠商對產業趨勢研判、本身核心競爭力與企圖心的不同，常在二到三年後漸行漸遠，留下非常多的受害使用者，辛苦無依的使用著不再有維護及新增功能的系統，
面對越益競爭的時代。有鑑於此，Oracle 應用系統運用自身的

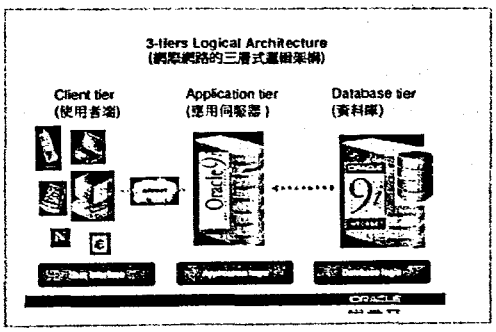
- 資料庫(Oracle Database)
- 應用伺服器(Oralce 9i Application Server)
- 開發工具(Oracle internet Development Suite)來建置、開發應用系統，從架構面解決長久以來存在的問題。

Oracle 系統採用三層式架構，提供企業級之服務。所運用之技術與產品都是業界標準，並廣爲使用。開放的平台和技術能有效的降低未來維護及管理的成本與效率。Oracle 架構及應用規劃分述如下：

- 資料庫服務層：採用開放性平台中最穩定、市佔率最高之 Oracle 資料庫，作為資料儲存平台，保障企業最寶貴之無形資產。並運作在最新型之 IBM eServer Pseries 690 上面。透過 TCP/IP 通訊協定與應用伺服器服務層連結與交換資料。
- 應用伺服器服務層：採用 Oracle 最新一代之應用伺服器平台 9iAS，它擁有業界佔用空間最小、執行效能最高的 Java 引擎，提供穩定及大量企業級使用者的存取需求。應用伺服器服務層除了透過 TCP/IP 與資料庫平台連結外，它也透過 HTTP 通訊協定與使用者服務層連結。
- 使用者服務層：我們只需要瀏覽器(Browser)即可執行 Oracle 系統，不需要安裝任何其他之 PC 端軟體或 Plug-In 程式。使用者服務層透過 HTTP 與 應用伺服器連結。
  這樣的架構，Oracle 稱之為公共產業服務架構，就像電力產業、自來水產業一樣，打開開關、轉開水龍頭即可提供服務；Oracle 系統的使用者只要點選瀏覽器，即可使用 Oracle 所提供的財務、採購、物料等相關

應用伺服器層 - Oracle Applications Architecture
應用伺服器層處理企業邏輯，它作為中介層與使用者端,資料庫通信.Oracle 9i Application Server (9iAS) 包括底下的 6 個Server 提供以上服務 ：

- ✧ Web server
- ✧ Forms server
- ✧ Concurrent Processing server
- ✧ Reports server
- ✧ Discoverer server (optional)
- ✧ Admin server

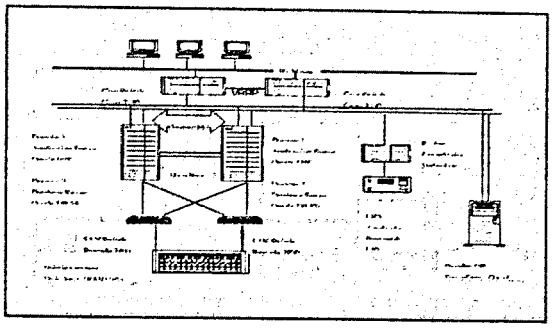


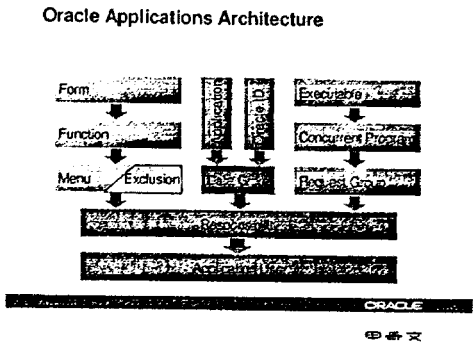
使用者端層 - Desktop TIer Architecture

使用者端包括使用 HTML 的自助式應用程式，使用 java applet 的 form-based 應用程式. 使用者機器可以根據不同的需求來決定是否安裝 Java run time environment , JInitiator.

系統後端需要一台資料庫伺服器(Database Server)和至少一台應用伺服器(Application Server)：
資料庫伺服器安裝 Oracle9i 資料庫和 GL 資料模型的資料結構，及 GL Server 端軟體。
應用伺服器安裝 Oracle9iAS 應用伺服器，包括運行 Discoverer ,Oracle Form 的 Server 端插件。

資料庫伺服器要求是 Unix 硬體平臺，GL 支援 Unix 硬體，包括 IBM、HP、Sun 的機器，對硬體型號沒有具體要求，應用伺服器建議是 Unix 平臺（也支援 Win 2000 Server），型號不限（IBM、HP、Sun），硬體設定 CPU 至少 2 個（主頻 >1GHz），記憶體 >2GB，硬碟 >50GB。Client 軟硬體需求： IE 5.5+ or netscape 4.6+.General PC with 256MB RAM.





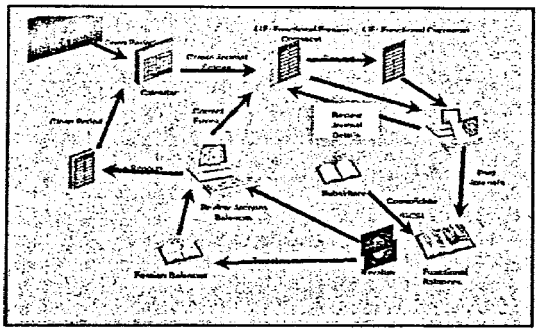
2-2 Infrastructure：網路環境、伺服器架構(AP/DB) Server-Client 軟硬體需求

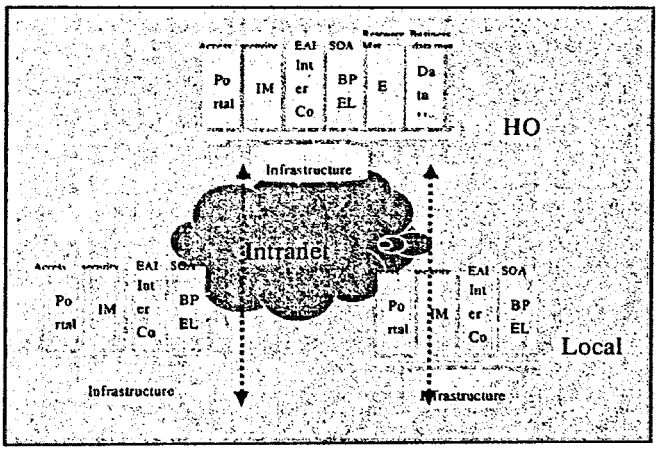
硬體結構爲 Browser/Server 和 Client/Server 的混合模式；對於前端 GL 業務使用者和系統管理員、報表管理員使用者需要相應的 Client PC 機，爲 Client/Server 結構運行環境。對於大量的資訊使用使用者只需瀏覽器接入 Intranet 即可，爲 Browser/Server 結構，如需要發佈資訊到 Internet，則需要設定防火牆等安全機制。

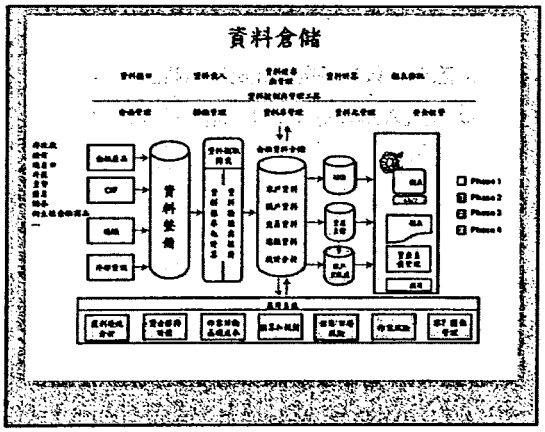
網路環境 Networking
Secure HTTPS network and TCP/IP connection
Wireless computing is designed by Leverage High-Speed Cellular Data Networks)

總部與地區之間的整合性系統架構

資料轉換技術：將舊有系統資料轉換至新系統的技術
系統穩定與效能：使用何種處理架構與技術增加系統運作穩定與效能，提供參考數據如頁面平均資料量大小、需求頻寬、頁面回應時間等。







3. 資料架構，資料轉換 及整合

3-1 資料庫管理

資料庫管理工具，金融業全行資料模型資料模型提供廣泛的管理功能。使用簡單而直覺式的視窗介面，系統管理員可輕易地進行以下管理：

- 提供建立及修改資料表、資料欄及資料列
- 可擴充系統定義與設計

3-2 伺服程序維護

- 檢視全部使用者所啟用之工作
- 終止某個使用者啟用之工作
- 清除伺服器系統資料表
- 單一用途對話框提供與金融業全行資料模型與系統間直接的溝通管道特殊對話框（召喚特殊識別），可組態、儲存供往後使用

3-3 金融資料倉儲架構

- 應採開放式關聯資料庫架構
- 與關聯資料庫為一體 (Data warehouse is the database)
- 允許進行資料庫資料結構客製化
- 金融業全行資料模型 - 專屬邏輯資料模型結構

資料倉儲：Data warehouse、Data Mart 與 Data Mining 技術

3-4 模型包括不同種類之資料表 (Table)

- ✓ 概要資訊表格(總帳/統計)
- ✓ 帳戶細目資訊表格(每種主要科目各一，包括存款、消費性放款、商業性放款、抵押、抵押保證、信用卡、投資等)及客戶相關管理

模型之結構

模型之結構應包括三個架構建立樹狀層次（層級）關係以利集合(aggregate)及drill-down，而每種架構又可以多種樹狀層次關係以利不同集合需求：

- ✓ 組織
- ✓ 產品
- ✓ 總帳會計科目

模型之資料管理工具

- ✓ 內嵌資料取出、轉換、傳送、載入工具 – 可先試載入
- ✓ 內嵌採礦技術 (Data Mining Tool)
- ✓ 唯一一套真實資料之資料庫無需為不同模組產生多個資料超市
- ✓ 內嵌 OLAP 工具
- ✓ 供使用者簡易產生報告工具
- ✓ 資料品質管制 –資料驗證- 驗證資料與控制品質 (日期)：
- ✓ 資料調整與校正工具
- ✓ 資料清除 (data cleansing functions)
- ✓ 資料集合 (data aggregation functions)
- ✓ 遵循 ODBC 標準之 SQL 語言之應用程式存取使用
- ✓ 標準的輸出格式/檔案形式如 MS Word/Excel 或其他標準的 (CSV) file
- ✓ 資料庫管理工具
- ✓ 資料庫安控工具
- ✓ 資料加密 (data encryption functions)
- ✓ 稽核追蹤以回存原始資料

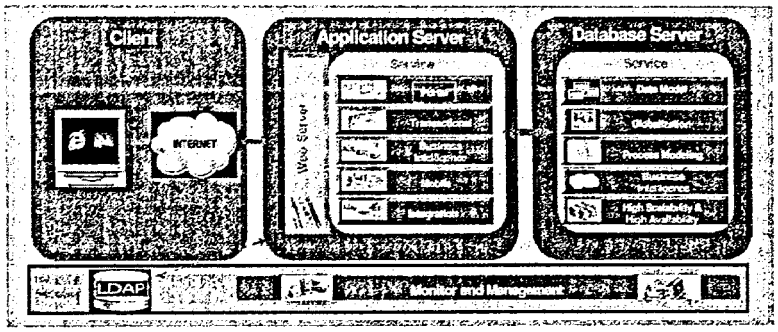
資料分享的方式

- 透過 SQL Net 分享利用 Oracle 資料庫 Client 端軟體，其他應用系統可以存取和分享 Oracle 資料庫；
- 透過 ODBC 方式：透過 ODBC 設定，其他應用系統可以存取和分享 Oracle 資料庫；
- 透過文字檔案：把 Oracle 資料庫內部導出到文字檔案，供其他系統使用；
- 透過 MS Windows 剪貼版：在 Discoverer 顯示的資料中，直接 Copy，然後在 Excel 中 Paste，可以直接交換資料；

3-5 會計系統系統架構

Server 與 Client 端操作平台：作業系統與資料庫類別；系統架構圖與建議 Server : Oracle can run on couple of platforms, including UNIX, linux, WinXP etc. Database: Oracle 9i
Client : IE 5.5+ or Netscape plus Java Plugin Jinitiator

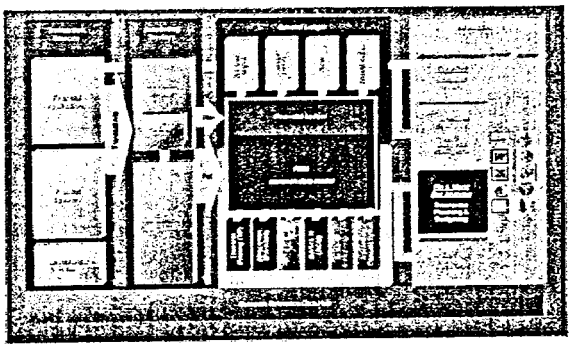
System Architecture : Oracle suggests the following 3-tier system architecture.

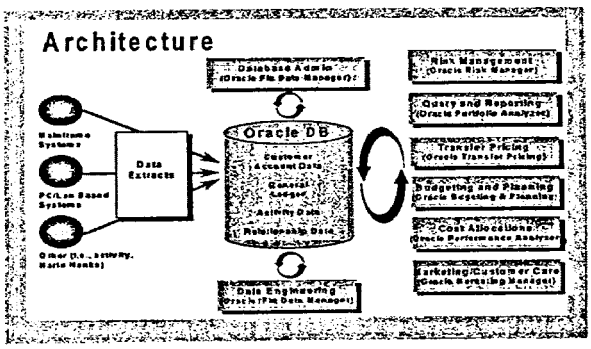


3-6 資料介面架構

System domain

- 交易會計循環 Tx/GL interfacing
- 雜項作業會計循環 PO miscellaneous
- 財務報告作業 Financial Reporter
- 績效報告作業 Performance Reporter

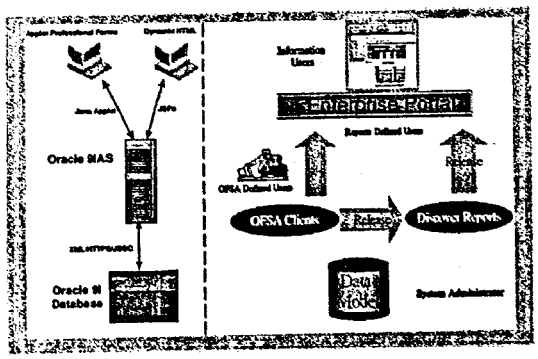




資料導出的能力和方式

- 透過 SQL 指令檔方式：可以直接編寫 SQL 程式，把 Oracle 資料庫資料導出到其他 Oracle 資料庫，透過 ODBC 或 Oracle 透明閘道技術導出到其他關聯式資料庫，導出到文字檔案；
- 透過 Oracle Export 工具：導出固定格式的 Oracle 資料庫文件；
- 透過 SQL Plus 工具：在 Oracle SQL Plus 互動式資料庫存取工具中，提供一系列的操作顯示資料格式化導出指令，產生標準格式的文字檔案；
- 透過 OEM 工具：使用 OEM（資料庫管理員），可以把資料導出到文字檔案、標準 Web HTML 文件
- 透過 Discoverer 工具：Oracle Discoverer 查詢工具，是 OFSA 應用軟體系統的結果查詢和報表工具，其由資料庫讀取的結果資訊可以導出到如下格式文件：

  - ✓ Microsoft Excel 工作簿：*.xls；
  - ✓ 超文字檔格式語言：*.htm；
  - ✓ 文字檔（Tab 定界）：*.txt；
  - ✓ CSV（逗號定界）：*.csv；
  - ✓ 格式化的文字檔（空格定界）：*.prn；
  - ✓ DCS（Oracle Express 格式）：*.dcs；
  - ✓ DIF（資料交換格式）：*.dif；
  - ✓ SYLK（符號連接）：*.slk；
  - ✓ WKS（Lotus 1-2-3 格式）：*.wks；



3-7 GL 資料介面處理的基本流程

OFDM 資料處理的基本流程

資料處理流程主要包括 12 個步驟，每個步驟的完成方法如下：

- 建立使用者自定的代碼和匯總表及標準的資料品質控制規則：由 OFSA 系統管理員和資料平衡與控制模組完成；
- 使用者自定代碼表的資料對照和導入處理：人工編寫程式完成；
- OFSA 產品交易彙總表的資料導入：建議先建立對應來源系統的中間過渡資料表（資料直接由來源系統得到，按目標資料表要求存放，未加工處理的資料），再透過程式進行資料轉換和加工處理，導入到 OFDM 產品交易彙總表中，程式可以是 SQL 指令檔、人工編寫或 OWB 控制程式完成；
- 總帳資料導入過渡表：由來源系統的總帳資料和統計資料導入到 OFSA 概要資訊表中，由於 OFSA 概要資訊表的導入必須使用系統提供的導入程式（儲存過程）完成，故需先安照系統要求建立過渡資料表，來源系統資料先由程式導入過渡資料表中；
- 總帳過渡表導入總帳統計彙總表 使用 OFSA 提供的總帳導入程式完成過渡表導入 OFDM 總帳統計彙總表的資料導入處理；
- 現金流量編輯器：爲保障應用（TP、RM）處理資料的正確性，必須對導入資料的現金流要素進行規則驗證，此功能由 OFDM 的資料平衡與控制模組完成；
- 總帳統計彙總表和產品交易彙總表的勾稽調整：對總帳統計彙總表和產品交易彙總表的差異進行勾稽分析與調整，保證資料的匯總對應關係和資料的一致與平衡；
- 資料加工處理：由 OFSA 的應用模組 TP、RM 完成；
- 處理結果資料轉換到報表輸出：由 OFSA 應用軟體完成；
- 概要資訊表轉入歷史資料：由編寫程式完成；
- 產品交易彙總表轉入歷史資料：由編寫程式完成；
- 產生使用者自定的匯總資訊表：由編寫程式完成；

OFSA 應用軟體系統的資料儲存在 Oracle 資料庫中，其資料的導入、導出和分享均依賴於 Oracle 資料庫平臺提供的功能，分別描述如下。

資料載入的能力和方式

- 透過 SQL 指令檔方式 可以直接編寫 SQL 程式 存取關聯式資料庫(Oracle、DB2、Sybase、SQL Server、Sybase）或文字檔案（txt 格式），進行資料的載入，非 Oracle 關聯式資料庫的存取，可以透過 ODBC 或 Oracle 透明閘道技術完成；
- 透過 Oracle Import/Export 工具：Oracle 的資料導入工具 Import，可以導入由 Oracle 資料庫 Export 工具導出的資料檔案；
- 透過 SQL Loader 工具：Oracle 的 SQL Loader 工具可以直接讀取文字檔案（txt 格式），進行批次的資料載入，OFSA Server 提供針對總帳統計表的資料載入範例（SQL Loader 指令檔），和對產品交易彙總表的資料對照、載入說明檔案；
- 透過 Oracle Warehouse Builder 工具：對於要求複雜的資料載入，Oracle 提供圖形介面的、功能強大的 ETL 工具－Warehouse Builder （OWB），可以定義多類型資料類型和載入方法，並控制元資料和載入過程，其支援的資料源格式包括：
  - ✓ Oracle 資料庫；
  - ✓ XML 檔案；
  - ✓ Oracle 透明閘道：主機（OS/390）資料庫、AS/400 資料庫、DB2、Informix、
  - ✓ Sybase、SQL Server；
  - ✓ ODBC：任意 ODBC 支援的資料源；
  - ✓ 文字檔案：含固定長度、定界字元、換行符的文字檔案，如*.txt、*.csv 等；

A9

Oracle Strategy in Financial System Architecture and Integration in Greater China

金融服務管理系統解決方案 及最優化戰略

Part II - Integration

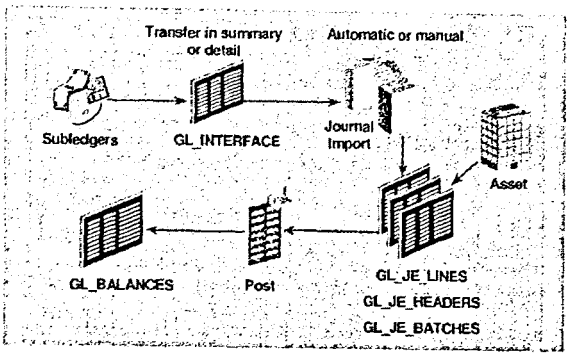
For the presentation to Oracle Financial Alliance in Greater China

ORACLE
甲骨文

2005年1月10日

Represented and Presented by:
Raymond Wu
Principle Consulting Architect

介面架構與整合技術



1 開放式整合方案類別

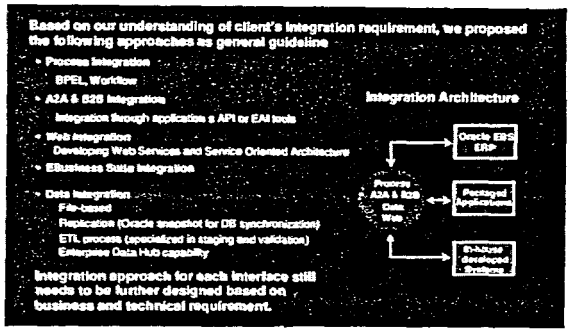
整合總帳系統 (Oracle 開放式平台) 與前端核心交易 (IBM mainframe 主機) 之間的資訊整合與外部不同平台 AP 的解決方案與使用技術, Oracle 開放式整合方案, 對於 IBM mainframe 資訊整合 具備以下數種方法;

- **Open Interface (file base batch data integration)**
- **ADI (file base batch data integration)**
- **API (real time data integration)**
- **跨接橋樑 Gateway (Oracle Transparent Gateways – r.t. data integration)**
- **配伺接合 Adaptor (Oracle InterConnect adapter – r.t. data integration)**
- **BPEL (Oracle business process integration)**

2 Oracle 建議

Oracle 建議採用 Open Interface 與 Interconnect Adaptor 作爲批次非同步以及即時同步的整合方式 以整合 Oracle 與 IBM mainframe 之間 的資料流與介面問題, Oracle 同時支援中文轉換

Oracle 有多種整合方式, 這些方式可以用來整合各種不同的資訊系統與 in-house 開發的應用系統, 其功能與方式分述如下.



3 介面整合程式之開發方式及使用工具

對於系統與系統間的整合，一直是許多專案或系統導入過程中都必需面對、而且對於不同資料來源是否需發展相關的專屬程式來進行資料交換，也大大的影響到整合的成本。

考慮到系統架構的複雜性以及為了降低整合的成本，Oracle 提供一組開放式介面(Open interface)來供其他系統資料轉入使用，只要來源系統能夠依循此一組介面標準，則 Oracle 的標準程序就能將之轉入系統標準的表格中，讓相關的處理作業得以存取，而對於資料的來源，最常見便是純文字檔，為了簡化文字檔轉入 Open Interface 的動作，Oracle SQL*Loader 提供了最簡便的設定方式，便能將指定的文字檔轉入到 Open Interface 裡，而且同一類資料僅需同一份設定即可重覆使用在後續資料的轉入作業，對於程式開發複雜度的減低以及開發成本的節省上，均有相當顯著的助益。若以資料屬性來區分，一般有逐筆交易方式(on-line, 或稱線上即時交換方式)資料和整批(batch)上傳這兩大類，然以資料交換的程式製作方面來說，Oracle 所提供的程式製作方式並無區別，僅在於設定的執行時間上有所區隔或是中介軟體的主動觸發能力。另外，在整批資料方面，尚可再概分為定時、多次性整批資料及不定期、一次性資料兩種；對於定時、多次性資料(例如每個月或每天只轉入一次者)，建議利用中介軟體 (例如 Oracle interConnect)來做為資料的交換媒介，此乃因此種資料的轉檔具一定的可重覆性，所以雖然在開發初期會有部份額外的開發成本，但長期來說，能夠節省許多人工介入的工作成本及避免掉可能的人為錯誤、無效率等因素；而對於不定期、一次性的批次資料(例如開帳資料、或將被汰換掉舊系統的初始資料等)，則因為資料轉入的有限性，基於整合開發成本及徒增系統複雜度的考量，並不建議透過中介軟體來做資料交換，而僅是由來源系統整理資料並轉碼之(大五碼 Big5 相容格式)，以供 Oracle 做一次性轉入。同於定時、多次性資料，逐筆交易性資料的轉入亦建議以中介軟體為媒介，其所節省的成本以及即時性的需求，更較此類批次資料轉檔來得更明顯。

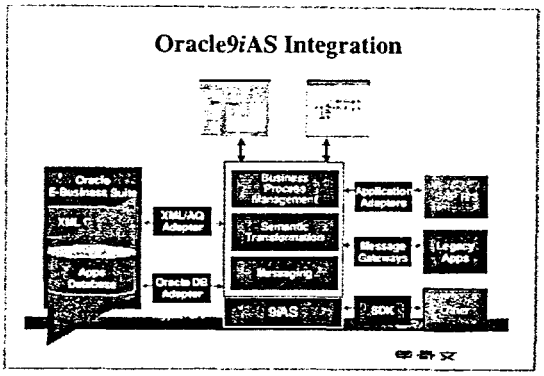
開放性介面 Open Interface/APIs

不論異質或同質系統與系統間的整合，資料交換格式的製定必需考慮到轉入資料的正確性和完整性，另一方面，也必需考量到不同系統來源資料讀取的方式；也因此，在整合作業進行中，單一系統對單一系統的整合點是首要的考量，可是一但整合的系統有數個以上，如何製定一個共通的標準介面，便是另一個相當重要而且能直接影響到整體整合成本的重要環節。

Oracle 中所訂定的Open Interface便是綜合整合成本、系統基礎的複雜度、轉入資料於系統中的完整性以及系統資料的安全性的考量，提供單一資料進入窗口爲資料整合點，只要依循此一標準介面，資料轉入後，系統程式便能將之分散轉入的基礎資料表格中，並能針對系統存取關鍵資料(Key Data)做調整及補入，以確保轉入資料在系統中的完整性。而任何需要轉入資料，便可透過此一共通的介面標準、選擇適檔模組的 Open Interface，便可與進行資料交換。使用這項工具，對於整合系統成本的考量而言，能有相當程度的幫助。 除了Open Interface之外，某些模組甚至提供了更快速的資料轉入方式，即是透過Open APIs；直接將資料經由API轉入比起Open Interface更直接有效，但相對的，這些過渡資料(Staging Data)就無法被保留在系統的某些地方，在錯誤處理的能力上，有一定程度的限制。

Open Interface 範例如下所示(圖示)，在固定資產模組處理固定資產主檔轉入的介面稱為"Mass Additions Interface"，下圖為部份欄位說明，節錄自 eTRM：

Oracle 9iAS interConnect and interConnect run time 圖型. Oracle 9iAS interconnect 是屬於中介軟體的一種.



Oracle 各系統模組中會提供開放介面,以提供 CUB 在系統第一階段建置時及建置後,與 CUB 未納入應用系統的資料連接。透過一致的開放介面可有效控制由非 Oracle 外部匯入資料的資料完整性。並可對匯入的資料作檢視、修改及重新傳送的作業。若 Oracle 資料需匯出時,也可透過開放介面集中匯出到外部應用系統。下列介面程式表中的介面程式表是經由與 CUB 使用者及資訊人員訪談後所歸納出來，在系統第一階段建置中須與CUB 未納入應用系統的資料連軌所需介面程式名稱及異動方式。

不論異質或同質系統與系統間的整合，資料交換格式的製定必需考慮到轉入資料的正確性和完整性，另一方面，也必需考量到不同系統來源資料讀取的方式；也因此，在整合作業進行中，單一系統對單一系統的整合點是首要的考量，可是一但整合的系統有數個以上，如何製定一個共通的標準介面，便是另一個相當重要而且能直接影響到整體整合成本的重要環節。Oracle 所訂定的 Open Interface 便是綜合整合成本、系統基礎的複雜度、轉入資料於系統中的完整性以及系統資料的安全性的考量，提供單一資料進入窗口爲資料整合點，只要依循此一標準介面，資料轉入後，系統程式便能將之分散轉入基礎資料表格中，並能針對系統存取關鍵資料 (Key Data) 做調整及補入，以確保轉入資料在系統中的完整性。而任何需要轉入資料，便可透過此一共通的介面標準、選擇適檔模組的 Open Interface 便可與其它系統進行資料交換。使用這項工具，對於整合系統成本的考量而言，能有相當程度的幫助。對於系統與系統間的整合，一直是許多專案或系統導入過程中都必需面對、而且對於不同資料來源是否需發展相關的專屬程式來進行資料交換，也大大的影響到整合的成本。考慮到 GL 系統架構的複雜性以及爲了降低整合的成本，Oracle 提供了一組開放式介面(Open Interface)來供其他系統資料轉入使用，只要來源系統能夠依循此一組介面標準，則 Oracle GL 的標準程序就能將之轉入 GL 系統標準的表格中，讓 GL 相關的處理作業得以存取之。對於資料的來源，最常見便是純文字檔，爲了簡化文字檔轉入 Open Interface 的動作，Oracle SQL*Loader 提供了最簡便的設定方式，便能將指定的文字檔轉入到 Open Interface 裡。

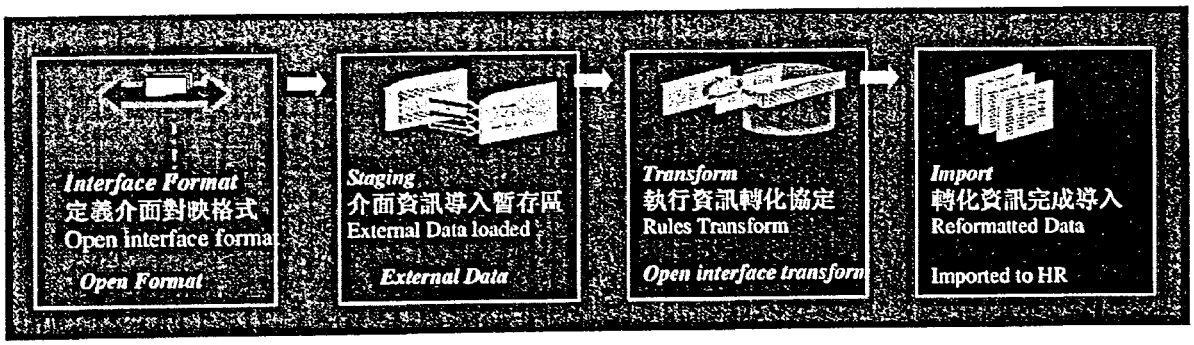
Open Interface 與 API 之間的比較

Open Interface	API
Use of Interface tables	No interface tables
Errors are stored in Error Tables	Errors need to be tracked
Need to run a import Process from interface tables To Application Tables	API populates original application tables directly
Concurrent Programs	PL/SQL or Java routines
Batch Processing	Real Time

Open Interface - file base batch data integration
與異值性系統間的整合趨向簡易
Easier Integration with Non-Oracle Systems
- Automatic data conversion that converts disparate data formats into an Oracle format
- Reapplication of the same rules each time you transfer
- Automatic validation on imported data

具備較大的調幅空間
Greater Flexibility
- Transform data from any source, regardless of the table structure of the source or target table
- Conditions allow you to control when Transformation Rules should be applied

提供資訊轉化規則 以便作執行暫存區資訊轉化協定
Rule Sets provide instructions to transform data in the GL_INTERFACE table:
- Use String Functions to parse and concatenate substrings
- Call PL/SQL Functions to perform sophisticated transformations
- Reference Lookup Tables to convert one value into another



Oracle IDT Data Integration and ETL process

IDT configuration option

4. Open Interface - Batch process

Oracle 提供了一組開放式介面(Open Interface) 來供其他系統資料轉入使用，只要來源系統能夠依循此一組介面標準，則 Oracle GL 的標準程序就能將之轉入 GL 系統標準的表格中.

5 Open Interface 資料同步作業 Synchronization

單一系統對單一系統的整合點是首要的考量，可是一但整合的系統有數個以上，如何製定一個共通的標準介面，便是另一個相當重要而且能直接影響到整體整合成本的重要環節。

6. API - real time data integration

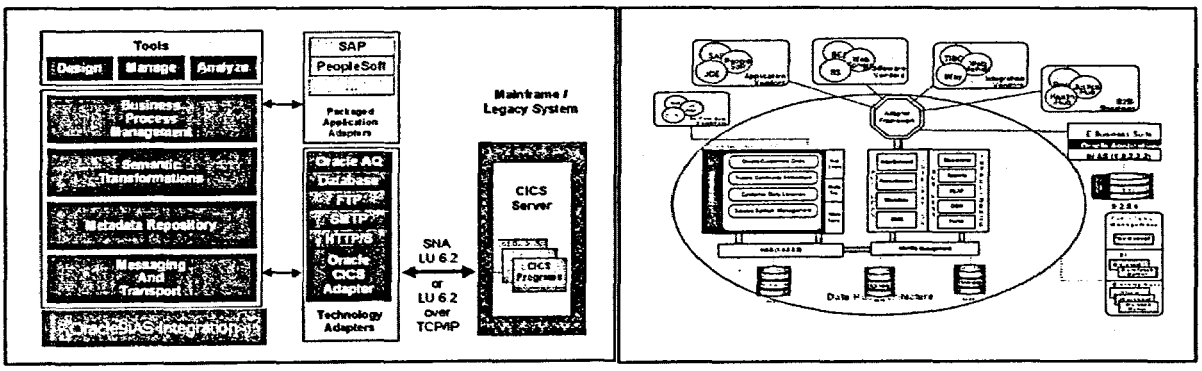
除了 Open Interface 之外，某些模組甚至提供了更快速的資料轉入方式，即是透過 Open APIs；直接將資料經由 API 轉入比起 Open Interface 更直接有效，但相對的，這些過渡資料(Staging Data)就無法被保留在系統的某些地方，在錯誤處理的能力上有一定程度的限制。

7. 跨接橋樑 Oracle Transparent Gateways

Oracle Transparent Gateways 是專門對非 Oracle 的系統介面提供跨接橋樑，對 DB2 的整合是建立 mainframe DB2 在 Oracle local 的 view，因而 Oracle 對 mainframe DB2 上的必要資訊具有透晰性. Oracle Transparent Gateways (TG4DB2) 對 mainframe DB2 的資料存取猶如執行 Oracle 本地的程序般的輕易，並將結果數值傳回 Oracle 資料庫.

配伺接合器 Oracle InterConnect adapter

Oracle 配伺接合器 整合 IBM mainframe CICS，使 CICS 應用軟體在 Oracle9iAS Integration 整合方案中形同 Oracle 本地系統的一項配備般的簡易操作，Oracle InterConnect adapter 以 IBM mainframe CICS 的用戶端點透過 SNA LU6.2 傳輸進入 CICS 泛用區(COMMAREA)，亦即 CICS 的資料進出交易區.Oracle InterConnect adapter 與 CICS 的 COMMAREA 共同形成 API 程序邏輯的嚴密架構， InterConnect adapter 簡易的用戶介面提供 IBM mainframe CICS 的執行選項以及資訊對映，其過程不需作任何 CICS 的程式設計，只需以定義方式讓 Oracle9iAS 自動將定義資訊存入接合器 Oracle9iAS InterConnect repository 中.

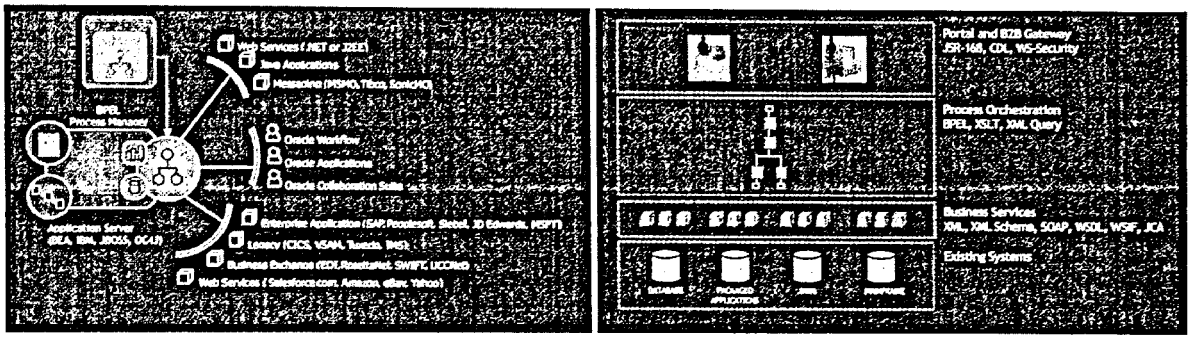


Oracle BPEL Process Integration

BPEL 又稱網路服務商業流程（BPEL for Web service, BPEL4WS），是一種以 XML 來描述企業內部流程的語言，使原本建立在不同產品上的商業流程也能像網路服務（Web services）可以跨平台互通。

中文系統 Chinese Font

Oracle 支援 Oracle 與非 Oracle 之間的中文轉換 IBM mainframe 的中文造字需有字形對應碼才能轉成文檔



8. 開發架構

Oracle 提供完善整合的交易應用軟體開發以及業務情報工具組合，可支援各種開發法、各種程式語言以及各種作業系統。 您的開發團隊可以更低成本更快速製作應用軟體，同時支援最新業界標準。 Oracle 開發工具可增加產能、簡化整合、提升應用軟體彈性，讓您快速回應多變的業務需求。 所有運用 Oracle Developer Suite 開發的應用軟體，都適合在 Oracle 基層架構配置的企業網格上運作。

Oracle9i 是電子商務應用程式和傳統應用程式的開發提供最佳的開發平臺。關鍵的開發領域包括：

- Entrise Java Engine
- XML 類型和 XDK
- SQL 和 PL/SQL 改進

Entrise Java Engine(以前被稱爲 JServer)

在 Oracle8i - 8.1.7 中，就已提供了一系列 Java 特性，Oracle9i Database 第一版對 Java 的支援主要集中在 :提高嵌入式 Java Virtual Machine(虛擬機)的性能 :提供新的 JDBC 和 SQLJ 功能 :增強 Java 存儲過程以及對 J2EE 容器（包括 Servlet Engine、JSP Engine 和 EJB 容器）的錯誤修正。

新的 JDBC 性能包括對 NCHAR 的支援以存儲 Unicode 資料（將物件類型的繼承關係揭示給 Java 開發者）、多級收集、Transparent Application Failover（透明的應用程式錯誤跳轉）、OCI 連接池以及基於 JDBC OCI 的中間層驗證、在 SQLData 表示中的 SQLJ Object 類型、本地化的 XA 支援、新的 LOB API 以及新的 Oracle 擴展的 oracle.jdbc.*介面。
新的 SQLJ 特性包括對 SQLJ 的動態 SQL 語句、從一個無類型的 ResultSetIterator 中存取資料、可選的 Oracle JDBC 代碼自動生成以及相應的 JDBC 增強和程式邏輯的支援。
對 Java 存儲過程的增強包括對返回多行記錄（REFCURSORS,引用游標）的支援。

XML 支援——XML 類型和 XDK

Oracle9i 以許多增強的資料庫操作爲特徵，這些操作通 過 SQL 將 XML 存儲在資料庫中並將傳統資料庫資料轉換爲 XML 。這些是支 持 eBusiness Integration（電子商務集成）、Packaged Application（打包的應用程式）和 Internet Content Management（因特網內容管理）聚焦領域所需的關鍵功能領域。 在 Oracle9i 中兩個主要 的 XML 支援領域是：

- 內置的 XML Developer Kit（XML 開發者工具 XDK）
- 本地 XML 類型

將 Java XML Developer Kit (XDK) 預裝到 Oracle9i 中並 將 C XDK 鏈結到 Oracle9i 中，開發者能夠輕鬆地訪問基 於 World Wide Web Consortium(W3C) 的功能，這些功能 在 Oracle9i 中生成、操縱、轉換和存 儲 XML 格式的資料。 在 PL/SQL 和 C++ 中也是可用的，XDK 提 供 XML/XSLT 分析 :XML Schema 處理器 :XML Class Generators :XML Transviewer Beans 和 XSQL Servlet ，從而提供基本的構件功能，以允許開發者快速地用 XML 來支援其應用程式。此外，對於要求通過其內容管理應用程式存儲和檢索大量複雜 XML 的開發者來說，Oracle9i 中的 XML 類型支援提供 XPATH 導航功能來優化性能。

SQL 和 PL/SQL 改進

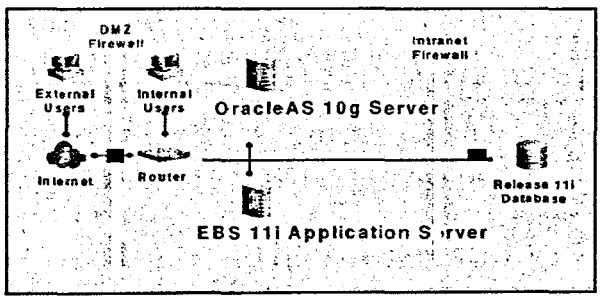
在 Oracle9i 中繼續改進 SQL 和 PL/SQL 以滿足現代開發的要求。除了本地編譯外，還改進了總體 PL/SQL 編譯。爲了增強物件的有用性，繼承、類型演變和動態方法分派在 Oracle9i 中均得到支 持——大大降低了在應用程式開發中使用物件的維護成本。Oracle9i 也支援新 的 ANSI 要求，包括對 CASE 語句、符合 ANSI 的鏈結和保留名版本標注的支援。爲了幫助從 非 Oracle 資料庫移植到 Oracle9i ，添加了滾動游標支援。
對 PL/SQL 的一些其他改進包括：

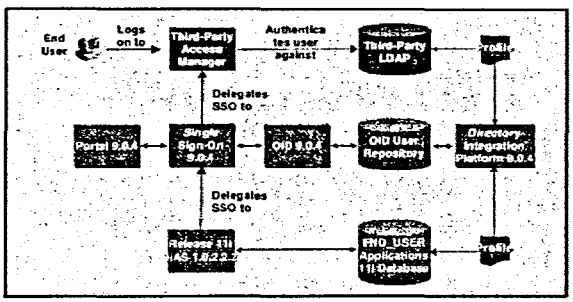
- 用於 SQL 編譯的集成前端——這意味 著 PL/SQL 直接支援所有的 SQL 語法更改（在嵌入到 PL/SQL 中的 SQL 中）。
- 將 SQL 並行查詢機制擴展爲用 3GL 語言（PL/SQL 、Java 以及外部例行程式）編寫的存儲過程。目前存儲過程能夠以流水線方式（遞增地返回）將資料傳遞到調用 SQL 語句。
- 對 ANSI 樣式的 CASE 語句和運算式的完全支援
- 通過本地編譯支援爲計算密集型應用程式提供了更好的支援。
- 將 SQL 和 PL/SQL 運行時引擎更緊密地集成在一起以提高性能。而且，降低了 從 SQL 調用 PL/SQL 過程的費用。

9. 系統安控

LDAP 權限控管、防火牆或其他安控方式
統合讓安全性達到最大化，同時能夠降低成本。 在今日經濟環境之中，企業都在尋找安全性政策，希望有助於達成業務目標，而不用增加複雜性與成本。 隨著 25 年以上的關鍵資料管理與基層架構經驗，Oracle 的安全性解決方案提供以下優勢：

- 讓多重社群的使用者能夠安全地共享統合資料
- 跨企業的整合式使用者供應功能以及存取管理
- 適用於 J2EE、網路以及既有應用軟體的單一登入功能
- 建立在開放式 LDAP 標準上的授權目錄儲存





Oracle 對存儲資料加密功能的增強包括將一個安全亂數產生器（RNG）合併到 DBMS_OBFUSCATION_TOOLKIT 中。在密碼系統中，安全密碼的產生是至關重要的一步：弱的、可預見或容易被人猜到的密碼會十分輕易地被破解。在 Oracle9i 中，由於能夠將所產生的亂數作爲安全密碼使用，這極大地促進了存儲資料加密的應用。

LDAP access control
Oracle Label Security 是一個精細粒度訪問控制產品。它將一個特殊的標簽添加到資料行中，提供複雜而靈活的行標簽安全。它建立在 Oracle9i VirtualPrivate Database（VPD）技術的基礎之上。Oracle Label Security 建立在政府和防禦機構用來保護機密資訊和提供資料分隔的標注概念的基礎之上。應用程式託管、保健和其他行業也可以利用資料標注來幫助滿足因特網時代的安全要求。例如，在應用程式託管中，預訂者標簽可用來分隔同一應用程式中的預訂者的資料。在資料庫中強制執 行 Oracle Label Security ，即使繞過應用程式也會提供安全。標簽提供了一種使用現有的應用程式資料不易達到的訪問控制範圍。Oracle Label Security 還包括一個複雜的策略管理工具，以管理策略、標簽和用戶標簽授權。Oracle Label Security 是通過因特網進行的 VPD 解決方

案。

資料加密 – 應符合產業認証之十二種方法以傳輸中及儲存資料時執行資料封裝加密，以防止帳戶詳細資料外泄露。必須具備以下資料安全性功能:

1. 支援 SSL 和 X.509 v3。
2. 以每個交談期 (session) 或每個物件為基礎的自動稽核檢查。
3. 必須能實施 Row Label Security 及 VPD (Virtual Private Database) 政策。
4. 具備支援多層次使用者代理驗證/授權 (N-Tier Proxy Authentication/Authorization)。
5. 提供精緻的稽核功能，可針對 Query 和 DML 指令進行監督。

依使用者角色設定安控權限範圍包括查詢、列印、修改、刪除資料權限；建立、修改、設定和刪除使用者安控權限範圍必須由兩位管理者獨立設定和認証。使用者有一個單一認証身分；
使用者依其認証身分必須輸入其特定密碼才得以進入系統。任何系統重要功能必須由兩位管理者獨立設定和認証。系統應具備稽核追踪報告。系統應產生錯誤報告。

以下將就三方面，使用者權限控管、功能權限控管與 資料權限控管三方面來說明Oracle e-Business Suite之權限控管，包含權限控管設定及使用者使用狀況稽核。

10. 使用者權限控管

身分認證
使用者必須透過個人ID及密碼登錄， 並由系統確認無誤始可登入系統;登入後使用者可看到所有經授權可使用的功能， 警示通知， 簽核通知， ...等 (請參考圖一及圖二)

使用狀況查核

透過監控畫面系統管理員可查詢使用者使用狀況， 包含使用者目前登錄的職責， 使用的表單已使用時間， ...等; 並提供相關報表供未來稽核與查詢使用
使用者監視器畫面

功能權限控管

(1) 功能授權 - 系統管理員可依使用者的工作職責與權限範圍， 指派合適的職責給使用者; 使用者登入後即可看到所有經授權之職責
使用者職責指派畫面

功能權限管理 - 系統管理員可依公司內使用者的工作內容與職責， 訂定不同的職責於系統內，每一職責可指派一組功能目錄，每一目錄可以包含不同的功能與報表; 在將適當的職責指派給相關的使用者， 每一使用者可同時擁有多個職責

使用者/職責/目錄/功能關係圖 (3) 使用狀況查核 - 系統管理員可針對重要資料啟動稽核功能 只要有任何使用者新增/更改/刪除所稽核之資料，系統將自動紀錄使用者 ID，更改前/後資料內容，更改時間 .等; 系統管理員或稽核人員可定期列印報表或查詢資料變更狀況

11. 資料權限控管

(1) 依資料/交易/欄位設定權限，系統管理員可依使用者權限設定細部資料權限控管， 例一： 採購人員有料號主檔的維護權限， 但只能維護與採購相關之屬性， 銷售人員也有料號主檔的維護權限， 但只能維護與銷售相關之屬性; 例二： 只要有授權之廠商皆可透過供應商入口網站(iSupplier Portal)進入系統查詢最新訂單及帳款資訊， 但只能看到個別廠商之資料 依使用者設定資料保護屬性

除此之外，Oracle Application 採用開放式的架構。所以，可以利用Oracle資料庫內的各種安全控管功能，來增強Oracle Application的安全控管機制。
(1) Internet電子商務有三項基本的安全需求：保護您的資訊以防止不當的竄改 — 這可由加密(encryption)的方式解決；證明顧客和零售商的確實身份 — 可由認證(authentication)技術而獲得

解決；確保任一方能作其想作之事 － 由權限設定(authorization)技術便可解決。Oracle 應用系統提供上述之三項安全需求。

Oracle資料庫經美國的TCSEC C2及歐洲ITSEC E3評估核定，並且能將密碼加密儲存，具備安全稽核（audit）功能，監控資料記錄在資料庫字典或作業系統檔案中及支援唯讀的表格空間。提供以角色(role)爲ANSI/ISO SQL3安全標準基礎的安全管理，以進行以群組爲單位的權限控制。
資料加密強化包括 Java Cryptographic Architecture (JCA) / Java Cryptographic Extensions (JCE)，支援大多數通用的加密與資料安全性演算法。Oracle Label

Security 是精心設計的存取控制產品，內建在Oracle9i Virtual Private 資料庫技術上，可在資料列中新增特殊標籤，提供周延且具有彈性的列標籤安全性。資料加密具備精密的存取控制， 支援SSL和X.509 v3。可選擇資料庫內建或外部作業系統的使用者驗證。以每個交談期(session)或每個物件爲基礎的自動稽核檢查。Oracle Internet Directory提供密碼架構的使用者管理、重試限制等密碼政策管理功能，以及適用於VPD 政策的管理工具。VPD 的應用程式內容也可以外部化，然後儲存在Oracle Internet Directory，以提供更佳的控管和使用者延展性。使用者密碼也可以運用標準或自訂編碼方法予以加密。

(2)結合LDAP，強化與外部系統整合
Oracle Application 可以利用 Oracle LDAP server
(Oracle Internet Directory-OID)儲存、管理 user credential。

12. 資料庫的安全性

OFSA 應用軟體系統的安全性，完全建立在 Oracle 資料庫平臺之上。
Oracle 資料庫在 C2 級的作業系統上(如商用 UNIX,VMS 作業系統)，不僅滿足 NCSC C2 級安全標準，而且已經正式透過了 NCSC C2 標準的測試。在 B1 級的作業系統上不僅滿足 NCSC B1 級安全標準，而且已經透過了 NCSC B1 級標準的測試，Oracle9i 目前也是唯一符合 B2 級安全標準的資料庫。

Oracle 資料庫的安全性包括系統權限，資料權限，角色權限三級結構。且角色形成樹形結構，且能動態生效或無效，從而完成動態的權限處理。所謂系統權限是指對資料庫系統及資料結構的操作權，例如創建/刪除資料表、資料索引、觸發子、資料連結、同義詞等。所謂資料權限是指使用者對資料的操作權，如查詢、更新、限制等，並可將存取權限控制在欄位級。所謂角色權限是把幾個相關的權限組織成角色，角色之間可以進一步組合而成爲一棵層次樹，以對應于現實世界中的行政職位。角色權限除了限制操作權、控制權外，還能限制執行某些應用程式的權限。角色還能動態地生效或無效，從而完成動態的安全控制。這樣的安全控制體系，使得整個系統的管理人員及程式開發人員能靈活控制系統指令的運行、資料的操作以及應用程式的執行，使資料機應用程式得到很好的保護。Oracle 還具有資料庫稽核功能，能對資料庫發生的所有修改進行稽核，還有事後觸發子，完成特定內容的稽核。

Oracle9i 繼續提供業界最安全的應用程式開發和部署平臺。關鍵聚焦領域包括：強壯的三層安全：透過代理認證增強了三層安全，包括 X.509 許可證文件或判別名 (DN) 的信用代理、對胖的支援、應用程式使用者的連接分享（胖 JDBC 和瘦 JDBC、OCI）和與 LDAP 的整合。一個可擴展的、安全的應用程式角色可以強制使用者只能透過中間層存取資料庫。結果是在應用程式的所有層中安全地維護使用者身份，將使用者和權限管理集中在基於 LDAP 的目錄中。

託管環境的安全：透過 Virtual Private Database (VPD) 增強特性、小粒度稽核和基於 web 的 single sign-on 提供託管安全。VPD 增強特性包括分區的應用程式上下文（取決於哪個應用程式存取資料的安全強制）和透過全局或分享應用程式上下文的連接分享。小粒度稽核用預設值變數提供了 SELECT 語句的選擇性稽核，基於"相關"列的存取，大大增強了每個使用者的可計算性。

A10

Oracle Strategy in Financial System Architecture and Integration

金融服務管理系統解決方案 及最優化戰略

Part III – Opeartion

For the presentation to Oracle Financial Alliance in Greater China

ORACLE
甲骨文

2005年1月11日

Represented and Presented by:
Raymond Wu
Principle Consulting Architect

The Operation Impact Categories overview

1. *Integration and Architecture*
2. *Data quality and Integrity*
3. *Business process and software package*
4. *System Performance*
5. *Web mgt, Security and Networking*
6. *Workflow; Alert, Warning & Reporting*
7. *System, Database mgt, Grid, HA & DR*

Summary

Oracle 提供業界所需之正確性, 一致性與展望性, 高度客製化架構特色讓業界毫無限制地決定資料處理深度、如何小計、如何總計及如何處理。這也是爲什麼 Oracle 仍是目前金融服務產業中最具彈性的資料庫的主要原因。

1. 系統管理

消除資料庫設計之不確定性
金融管理資訊產業裡最具彈性、功能性之資料模型。具備遠離觸礁、與報表產生應用程式，亦可依照您的需求完全客製化。而其亦可利用 ODBC 相容工具進行擴充與連結。

完整及全面性的維護資料
包含詳細資料（交易及客戶帳戶細目皆有）與總覽金融資料。因此細目資料可集中供任何模型所需，所有金融決策支援系統亦可使用相同之資料。

快速及彈性地存取資料
採用開放架構之關聯式資料模型。允許建立自訂資料欄位與表格，並為任何資料欄位選取使用者自訂值。此外，亦可讓任何採用遵循 ODBC 標準之 SQL 語言之應用程式存取使用。

資料庫管理
提供建立及修改資料表、資料欄及資料列以符合您的需求，讓您將資料庫之結構變更爲符合預期要求。提供資料庫最佳化工具，同時亦可擴充系統定義與設計。

伺服程序維護
系統管理員：

- 檢視全部使用者所啓用之工作
- 終止某個使用者啓用之工作
- 清除伺服器系統資料表
- 使用兩種類型之工具 – 單一用途對話框提供與資料模型與系統間直接的溝通管道；而特殊對話框（召喚 ID），可組態、儲存供往後使用。

資料管理員 - 平衡與控制
資料完整性與強化資料之控制爲金融效率管理之關鍵。要讓資料有價值，必須讓資料有效，資料品質必須得到控制，還有必需能夠調整。否則，有可能根據衝突之假設做出決策。
平衡與控制爲一資料驗證、資料調整、資料清除與校正工具，用於消除此類衝突與顧慮。其驗證您的資料與控制品質。當發現資料不一致之情況時，允許即時更正，調整一般帳務加總與標準控制號碼之細目資料，提供重要的反饋更正相關系統中之錯誤資料，其設計用於編輯與管理帳戶階層與不同集合階層之資料，可使用預先建立之集合爲：
評估歷程資料內是否不再需要之詳細資料，如帳號、交易量等；
利用將相同現金流特性（如起始日期、付款週期等）之細目資料群組集中，強化現金流；
可供多重使用者存取使用之預先建立報表集合：
藉由集中客戶資訊及收益資訊，評估特定商品（存款、抵押等）之客戶收益，並可進行訊分析與模型化。

線上輔助的完整性和易用性
應用軟體系統提供的線上輔助，包括如下方式：
介面上的動態操作提示資訊：在應用軟體系統的介面底部一行，是動態操作資訊提示行，隨滑鼠移動位置變化，即時動態顯示操作提示資訊，快速指導使用者操作；打開每個 ID 時，點擊滑鼠右鍵，獲取該 ID 的提示資訊：因爲所有操作均歸結爲相應的 ID，在打開每個 ID 時，點擊滑鼠右鍵，即可獲得該 ID 的背景提示資訊，輔助使用者快速查閱相關資訊，資訊提示示例如下：

透過 Help 功能表，獲取基於 Web 的上網輔助資訊：點擊 Help 功能表，可快速選擇查詢輔助資訊，包括資訊檢索、快速定位查詢等，透過 Web 顯示豐富的上網輔助資訊；在系統的每一步操作均提供線上輔助，螢幕底層提示行是簡介明瞭的操作功能提示，透過 Help 功能表查詢詳細的、全部的上網輔助檔案。

上網輔助檔案是完全 Web HTML 格式的，並提供多種資訊檢索方式，其內容包括了全部的應用軟體的使用者手冊和技術手冊，是非常完整和易用的。

資料備份和恢復
應用軟體系統的資料，全部儲存在 Oracle 資料庫中，Oracle 9i 資料庫提供了強大的資料備份和恢復能力。
Oracle 提供多種備份與恢復機制，不同的備份機制具有不同的目標和方式，從而相應的恢復也是不同的。它們有簡單有複雜，同時恢復的程度也有差別。可以以這二種方式來劃分： 物理備份和邏輯備份，它們最主要的分別是邏輯備份是採用 Oracle 資料庫的工具（如 EXP/IMP）進行備份，而物理備份是以作業系統的文件備份工具把 Oracle 的資料檔案拷貝，或者使用 ORACLE 的 RMAN 備份工具。物理備份又可根據備份資料庫時資料庫是否爲開啓（open）來劃分爲熱備份和冷備份，使用者可 以根據自己的實際系統運轉情況或
業務需求綜合各種備份 方式，提高系統的可靠性。或者也可以透過系統對資料安全性 的要求，確定必要的備份機制，反過來再確定 Oracle 資料庫系統的運行方式。

本地化支援能力和客戶化修改問題
應用軟體因其應用目標和針對使用者的不同，系統全部支援中文的輸入、輸出和顯示，但針對不同應用目標和使用者的特定要求，部分應用的操作介面爲英文，對於不同應用和使用者的中文支援程度如下：

針對資訊使用使用者的查詢和報表應用 Discoverer：系統的最大量使用者群，約占 80%以上的使用者只是查詢和使用管理會計的結果資訊，此功能透過 Oracle 查詢工具 Discoverer 完成，最終使用者端爲全中文的操作環境，包括功能表介面、輔助資訊、上網檔案等，使大量的資訊使用者快速理解和使用，在 Web 環境和 Client/Server 結構下，都提供全中文支援，包括操作介面、提示資訊和輔助檔案。

應用軟體針對資產負債管理部門的業務操作使用者，由於此軟體的專業性和國際銀行界通用術語的限定，此軟體的操作介面功能表爲英文，系統附帶的標準檔案（包括輔助資訊和操作手冊等）均爲英文，但所有使用者定義的條件、參數項和相關域，全部支援中文的輸入/輸出，而且對於此類高素質使用者而言，採用國際通用的標準，對操作資訊的理解將更爲準確，系統功能表介面由於採用了大量專業術語和簡略語，在目前尚沒有統一的中文標示標準情況下，硬性的字面翻譯成中文，可能會引起歧義和對業務人員的誤導，應用軟體
軟體在全球的 150 多個客戶，包括亞洲許多國家，及香港、臺灣地區銀行客戶的導入中，對於此類專業操作軟體的英文介面均表示理解和接受，故我們建議銀行考慮此實際情況，對於操作提示資訊和輔助檔案，可以在必需情況下翻譯成中文；系統管理員應用：鑒於此類使用者的管理責任和技術要求，大部分應用介面爲英文，資料庫提供全部的中文支援（介面、檔案等），系統管理應用爲英文環境，但全部支援中文的輸
入/輸出和顯示；

對於軟體的修改和客戶化問題，首先要定義什麼是客戶化， Oracle 應用軟體系統作爲一個成熟的銀行應用管理軟體而言，按照客戶的需求和客製是透過系統導入過程中的流程設計和參數設定來完成，這些客戶化的要求，我們不稱之爲系統的更改，而是標準的產品參數設計設定和導入過程。

如使用者對產品功能提出特殊要求而需要系統級的更改，則需提交 Oracle 軟體研發部（美國）申請處理，可能會發佈針對特定使用者的額外的修正程式，時間需要考慮變更需求的範圍和影響程度而定；對於國內商業銀行，變更的次序應爲政府規定要求的變更、銀行特殊需要的變更、爲適應市場的變更、保持相容性的變更，因爲作爲銀行管理資訊系統的應用軟體，一旦導入完成後，

只要滿足所需功能，一般情況下不會改變系統平臺（包括改變硬體和軟體平臺、應用軟體版本等）。

系統的開發性和開放性

應用軟體系統是完全基於 Oracle 技術平臺完成的，包括 Oracle9i 資料庫、9iAS 應用伺服器和 9iDS 開發工具包（含 Warehouse Builder、Discoverer 等），基礎模組－資料模型，實際上是一系列預定好的資料庫目標和儲存規則，對於外部資料的輸入與處理，及系統的處理性能支援，皆依賴於 Oracle 資料庫和相關工具。結果資訊和報表的 Web 發佈，依賴於 Oracle 9iAS 應用伺服器的支援，Oracle 技術平臺的優異性能和廣泛的適應性，爲應用軟體的完成提供了強大的技術支援。

改錯操作的簡便性和步驟

對於輸入資料中發現的錯誤修改處理，提供多種方法完成：在資料平衡與控制 B&C 模組中，定義對現金流欄位的自動修改規則：對於帳戶中涉及到現金流計算相關的欄位，可以在 B&C 模組中，定義如發現某欄位值錯誤的預設處理方式，如強制預設值等，可以在現金流驗證過程中，自動的對錯誤進行批次修正；在應用軟體系統中，採用資料驗證定義改錯：對於系統管理員權限使用者，可以在資料平衡與控制 B&C 模組中，定義資料驗證定義，可選擇線上查看或線上修改資料，可對每一記錄行的非主鍵索引欄位進行人工修改，適用於個別記錄錯誤的修改；系統管理員在資料庫管理環境中批次修改資料：系統管理員可以登錄資料庫 SQL Plus 環境，人工編寫 SQL 指令，對相關資料進行逐條或批次修改。

資料修補或調整的簡便性

對於業務系統中缺少的資料，允許使用者自定規則進行修補，方法包括：資料平衡與控制 B&C 模組中，定義對現金流欄位的自動修補規則：對於帳戶中涉及到現金流計算相關的欄位，可以在 B&C 模組中，定義對來源系統缺少資料的預設處理方式，如強制預設值等，可以在現金流驗證過程中，自動的對資料進行批次修補。

在應用軟體系統中，採用資料驗證定義修補：對於系統管理員權限使用者，可以在資料平衡與控制 B&C 模組中，定義資料驗證定義，可選擇線上查看或線上修改資料，可對每一記錄行的非主鍵索引欄位進行人工修改，適用於個別記錄的修補；系統管理員在資料庫管理環境中批次修改資料：系統管理員可以登錄資料庫 SQL Plus 環境，人工編寫 SQL 指令，對相關資料進行逐條或批次修補。

在資料 ETL 工具中，定義修補規則：如在資料導入的 SQL 程式，或 OWB 環境中，定義對缺少資料的修補規則，如強制設 0 等，在資料導入時，自動批次完成資料修補。

外部市場資料和外部系統的接入

對於外部市場資料，可以透過讀取文字檔案，或直接存取外部相關系統的方式完成，但都需要在導入過程中，開發相應的介面程式。對於外部系統，不提供對任何外部系統的即時對接功能，如客戶需要，可以透過開發相應程式完成，但介面程式的開發涉及到外部系統授權，需要系統整合商協助，統一對外介面的開發工作。

系統處理速度

借助於 Oracle 技術平臺，能夠支援大資料量、大使用者量的高性能處理，並發存取的使用者量可達到數十萬，系統的處理性能可參考如下的實際案例測試結果：

客戶名稱：美洲銀行（Bank of America）；
硬體：使用 12 個 CPUs 和 48GB 記憶體的機器。
資料：380,000,000 交易記錄，120,000,000 個帳戶性能指標：

利用 Transfer Pricing 模組計算 97,500,000 筆記錄的時間爲 3.7 小時變動交易的處理時間爲 26 分鐘信用風險和市場風險的處理時間爲 75 分鐘處理存款交易表（300,000,000 記錄）的時間爲 2.75 小時處理貸款交易表（40,000,000 記錄）費用交易表（40,000,000）的時間爲 44 分鐘

Oracle9i Entrise Edition

如同 Internet 正逐漸改變我們使用和存取資訊的方式，Oracle9i 亦使資訊管理的方式爲之改觀。Oracle9i Entrise Edition 是特別針對支援 Internet 和傳統的關鍵任務 OLTP 及資料倉儲應用系統之需求所精心設計的。從開發過程到完成後的配置階段，Oracle9i 提供了無以倫比的創新功能，提供管理企業 Web 化過程所需的全部資料。

Oracle Enterprise Manager

Oracle9i 提供了一個包羅萬象的管理工具—Oracle Entrise Manager，用以管理 Oracle 的資料庫和應用環境。Oracle Entrise Manager 包括一個易於使用的集中式主控台，多元化的管理工具組、以及偵測和解決任何可能發生問題的延伸能力。它亦包含數種管理方面的應用程式，以執行資料庫和應用系統的日常工作，例如例行的備份作業之排程。Oracle Entrise Manager 是一個涵蓋範圍極廣的系統管理產品系列，並提 供對協力廠商及客戶應用程式的開放式界面，包括與所有主要的網路管理平台之整合。

此一產品系列包含新一代的系統管理工具，其設計目的是爲有效的管理完整的 Oracle 環境，包括系統、資料庫、網路、和應用系統，以減少管理人員必須花費在管理複雜的生產環境所耗費的時間。

Java 管理工具

Java 已迅速成爲現今最受歡迎的程式語言，這是因爲其特質對每個人都深具吸引力之故從發現它是一個強大的物件導向語言的開發人員，到認爲它可降低成本並簡化應用系統之開發與維護的 CIOs 和經理人皆是。Oracle9i 將一個可延展的 Java Virtual Machine (Java VM)與資料庫伺服器相整合，而進一步延伸 Oracle 對 Java 的策略。Oracle9i 亦支援 SQLJ，一個可將 SQL 內崁於 Java 程式碼的語法。Java 與資料庫之整合可讓程式開發人　員根據開放的業界標準，建立高效能、可延展的應用系統。

在 Oracle Data Server 中的 Java VM 使開發人員能夠在資料庫中撰寫、儲存、和執行 Java 程式碼。這表示他們能使用 Java 語言來建置預儲程序、功能函數，或是觸發器。在資料庫伺服器內的 Java VM 亦可戲劇性地降低撰寫資料庫程式所需具備的技能。相對於 C、Oracle Call Interface、PL/SQL，和其他的語言，開發人員現在僅需單純地使用 Java 便能建立完整的應用系統。SQLJ 提供關聯式資料庫與 Java 之間一個強大且易於使用的介面。它是由 Oracle、IBM 和 Sun 所共同發展出來的開放式標準。利用 SQLJ，SQL 敘述句便能輕易地內崁於客戶端或伺服器端的 Java 程式碼中，以便與關聯式資料庫相互溝通。客戶端的 Java 程式碼與 SQLJ 敘述句是透過一個 Java 前置編譯器，來產生以 JDBC 爲基礎的 Java 可執行程式。伺服器端的 Java 程式碼則可經由原始的介面來直接存取資料庫。

Oracle9i 亦支援 Entrise JavaBeans 以及經由 CORBA 通訊標準 —IIOP 來進行溝通。分散式 Java 物件能經由原始的 CORBA 協定直接連到 Oracle Data Server 中的 Java 物件。 JDeveloper 是 Oracle 下一代以元件(component)爲基礎的 Java 開發工具。JDeveloper 可讓開發人員建置完整的 Oracle 和百分之百的 Java 商業解決方案。JDeveloper 可支援從簡單的 JavaBeans 元件到泛企業應用系統的任何一種開發形式。 Oracle9i 亦與 Oracle Application Server 緊密地整合。後者提供一個開放式、以業界標準爲基礎的架構，用以開發和配置真實世界中的業務和商用應用系統。其可延展的分散式架構和優越的資料庫整合是支援關鍵性商業、以交易爲基礎之應用系統的根本所在。

Oracle WebDB 開發以 Web 爲基礎的應用系統

Oracle WebDB 是建立、配置，和主動監控 web 資料庫應用系統及內容驅動之 Web 網站的解決方案。藉由結合一個直覺式的 HTML 介面和一組完備、強大且以瀏覽器爲基礎的工具，Oracle WebDB 讓使用者能輕易而快速地開發 Web 資料庫應用系統。Oracle WebDB 可說是讓 Oracle 資料庫具備 web 能力的最快速且最容易的解決方案。Oracle WebDB 含有建置動態的 Web 應用系統和內容驅動的 Web 網站所需具備的全部工具。

利用 Oracle WebDB 所開發的應用系統與 Web 網站是完全包含於 Oracle 資料庫中。開發及配置 Oracle WebDB 應用系統所需的唯一客戶端軟體是 Web 瀏覽器。您無須在所使用的每部機器上安裝軟體、無須利用 FTP 在遠端管理檔案、亦無須每次在開發人員修正錯誤並將系統升級時規劃一套複雜的配置計畫。Oracle WebDB 能真正面對 Internet 所加諸於 IT 組織的挑戰— 快速的應用系統開發和簡化的配置作業。

Oracle Entrise Manager Console 是執行所有管理工作的主要介面。主控台的圖形顯示畫面包含了選單、工具列(toolbar)、和四個可客製化並可延伸遍及全體，結構的窗格(pane)。這些窗格包括：領航員(Navigator)、群組(Groups)、事件系統(Event System)，和工作系統(Job System)。所有的 Oracle 管理應用程式都可從主控台中啓動。

Management Server 處理所有的系統管理工作以及這些工作的跨企業分派之管理，並提供跨系統的延展能力。Management Server 利用 Entrise Manager Repository 作爲其永久的後端儲存庫。此一 Repository 保留了系統資料、應用程式資料，以及分佈遍及整體環境之被管理物件(managed entity)

2. 智慧型代理程式 Intelligent Agents

Intelligent Agents(智慧型代理程式)是獨立執行於被管理節點上的處理程序。此代理程式負責服務探尋、事件監控、和工作執行。Intelligent Agents 亦支援 SNMP(Simple Network Management Protocol)，可讓協力廠商之應用系統能與代理程式相互溝通。SNMP traps 可傳送至 SNMP 主控台，如 HP OpenView。代理程式亦提供直接存取 Oracle 資料庫管理資訊基本變數(Management Information Base variables)之能力。

完善的備份復原計劃 (Disaster Recovery) 使用線上備份

針對 Oracle 系統的可用性，Oracle 提供線上備份機制，使用 RMAN（Recovery Manager）工具或監控軟體（Oracle Entrise Manager，OEM），控管軟體讓系統在備份的當時，線上的使用者仍然可以使用，達到全天候、全年無休的作業系統。完備的備份規劃因應回復需求一個完整的備份方式應該包含以下備份物件：

(1) 提供一個完整的作業系統 AIX 備份（IBM 人員配合事項）
(2) 備份 Oracle Application 11i 程式部分及軟體部分
(3) Oracle Application 11i 資料庫部分
(4) Oracle 備份資料庫的備份記錄（RCAT）部分 (註:若使用 RMAN 來備份)
(6) 監控軟體（OEM）備份

Oracle 對於系統的備份提供一系列完整的工具與規劃，以上物件分成兩類：一是資料庫部分，二是檔案部分。

資料庫部分

使用Oracle RMAN，以線上備份方式，來備份正式作業系統、品保系統、開發系統及教育訓練系統的資料庫部分，而線上備份(可搭配作業系統所搭載之備份工具)或者搭配Third party之備份軟體,(如 Veritas 備份軟體，使備份資料的保存更完善更快速)，而線上備份可以選擇完整資料庫備份，或是針對資料有異動部分來備份。提供Export工具來備份資料庫的備份記錄（RCAT），提供備份的另一個選擇可以選擇只將一資料部分備份出來。

圖表:備份規劃藍圖

備份目標	分類	備份頻率	備註
正式作業系統	資料庫	每天	線上備份
Oracle 應用程式（$APPL_TOP&$JAVA_TOP）	檔案	導入時期建議每天 上修復程式後備份一次 上線一段時間後則改成每個月備份一次	
備份資料庫（RCAT）	資料庫	每天	邏輯備份（Export）
Oracle 軟體	檔案	每季或上修復程式後	
開發系統	資料庫	每天	如果空間允許
教育訓練系統	資料庫	每天	教育訓練時期

完整的回復機制與演練
當系統因爲硬體毀壞或遇不可抗拒的外力因素時，需要一套完整的系統演練計畫。因此，在日常運作階段，即可提供備份磁帶回復演練，以幫助確保備份完整無誤。其先決條件在客戶端需提供設備做完整系統的複製（配合事項）。回復方針與案例：

2-1.復原軟體部分
主機作業系統毀損，通知硬體廠商將主機回復。
管理監控主機
將 Oracle 軟體回復，將備份記錄（RCAT）資料庫回復即可。或者重新安裝新的 Oracle 企業版本（Entrise Edition）及 Oracle Entrise Manager Create 資料庫 RCAT，將資料匯入 RCAT 資料庫。

2-2.復原資料庫部分
資料庫復原的方式需根據資料庫毀損的狀況而定，資料庫程序毀損的愈嚴重復原程序愈複雜，因此管理者需小心處理，Oracle 提供回復方式有：
完全回復到毀損的時間點。
復原到某個時間點。
復原到某個資料庫易動點（SCN）。
復原到某個交易易動序號（Log）。

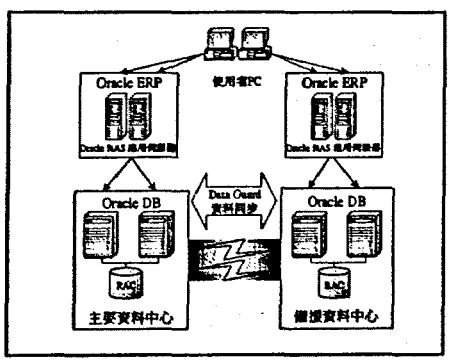
3.多重模擬毀損劇本

復原資料庫，且控制檔（Control File）及 Redo Log 未損毀。
復原資料庫，且控制檔（Control File）損毀但 Redo Log 未損。
復原單一個 Tablespace，且其他的檔案皆完好。
資料庫關閉後無法開啓爲 Open 狀態，只有 Instance 起來，其他的 Tablespace 無問題發生。
復原資料庫到某個時間點，或 SCN Number 或 Log 序號，且控制檔（Control File）未損毀。
復原完全毀損的資料庫。
復原 Recovery Catalog 資料庫。建議未來可採用的最高可靠性架構
在此架構下，使用者運用主要資料中心的資源進行作業，因為所有設備都有兩組，所以任何設備的故障都不影響正常作業。為避免因災害造成停機，故另外建置完全一樣的備援資料中心，當發生重大災害，導致主要資料中心無法立即修復時，則可利用備援資料中心繼續提供服務。
以下分別針對此架構中的元件加以說明：

應用程式層
Oracle 系統中所有的功能都在Oracle 9iAS應用伺服器上執行，配置兩部Oracle 9iAS應用伺服器除可避免單點失誤外，也可運用Oracle 9iAS叢集的特色，提供負載平衡和當機接手的功能。在備援資料中心則保有一樣的架構和完整的應用程式，以達到應用程式的異地備援目的。

網路設備
所有的網路設備（Router、Switch）都應具備容錯能力，並能自動將流量轉向，至其他可正常運作的路徑。以下架構提供最高可靠性的網路組態：
Tier 1：提供網路連線的容錯功能，以保證所有的使用者都能透過網路連線至資料中心。

Tier 2：提供應用伺服器層的容錯和負載平衡功能，並且提供單一的IP讓使用者連線。負載平衡設備負責接收所有的使用者連線需求，然後才依照負載平衡原則，將使用者連線需求轉向至最適合的應用伺服器。若有應用伺服器故障時，也由負載平衡設備將其餘使用者連線需求轉向至其他可用的應用伺服器上。



Tier 3：建置額外的Router。當主要資料中心無法提供服務時，能將使用者的需求轉向至備援的資料中心，繼續維持作業正常。

4. 資料庫層

Oracle 能利用資料庫叢集技術 （Real Application Cluster，RAC），將資料庫的可靠度提昇至最高。RAC 利用多台主機共同存取同一份資料庫檔案，讓使用者不論由哪一台主機存取，都可獲得一致的資料，以避免單點失誤的發生。除高可靠度的特性外，RAC 也能大幅提昇系統擴充性，藉由增加叢集中的節點，而不斷的增加系統的處理能力，在增加的過程不需要修改現有的應用程式，也不必重新組織資料。RAC 也能充分利用叢集資料庫，達成負載平衡和跨主機平行處理的功能。

異地備援
爲避免重大災害導致系統服務中斷，在地理上的遠端建立另一套環境是必要的，但是硬體設備的複製很容易，如何將兩邊的資料有效率的維持一致，是異地備援方案中較困難的部分。Oracle Data Guard 是利用軟體技術，即時將主要資料中心的異動傳至備援資料中心，維持兩邊資料一致的解決方案。其在網路上傳遞的資料是以 Log的方式傳送，對於網路頻寬的要求較低，同時兩點間的距離也較不受限制。當災害發生時，可馬上將備援資料中心提升爲主要資料中心，即可馬上會復原正常作業。標準作業程序（Standard Operation Procedures，SOP）不論系統因何種原因故障，都應盡速加以排除，以回復到系統正常時所能提供的處理能量，因此制定完善的復原標準作業程序是絕對必須的。而此標準作業程序必須隨系統架構變更時，同步修正，並定期加以演練，以期能在狀況發生時，以最快的時間完成復原動作。

Oracle Entrise Manager

Oracle Entrise Manager 是一個整合型的系統管理及監控解決方案，可完整掌握整個 Oracle 執行環境，讓 Oracle 保持高可靠度和高效能。對於 CUB 資訊部門來說有許多的元件必須被監控與管理，包括：主機、資料庫管理系統、應用伺服器軟體等，維持這些元件的正常運作和良好效能是系統管理者重大挑戰。Oracle Entrise Manager 是一個整合型的系統管理及監控解決方案，可完整掌握整個 Oracle 執行環境，讓 Oracle 保持高可靠度和高效能。 現實環境中不可能配置專責人員 24 小時不斷的監控系統，因此整合於 Oracle Entrise Manager 中的 Oracle Diagnostics Pack 是最好的方案。Oracle Diagnostics Pack 的特色如下：

全天候的監控系統
可以不斷的收集資料庫上的效能指標、使用空間、記憶體快取效能；主機上的 CPU 使用率、記憶體使用率、磁碟空間使用率等資訊，並將這些資訊以圖形見面呈現給管理者，作為判對系統健康狀況的依據。

問題示警與自動排除
Oracle Diagnostics Pack 除了被動的收集數據給管理者外，也可自行設定監控條件，當數據超過零界點時，主動以電子郵件或呼叫器通知相關人員，若搭配自動修復程序，更可將影響層面減至最低。例如當資料庫服務監聽器意外終止時，除馬上通知管理者外，可透過自動啟動程序馬上重新啟動資料庫服務監聽器。現實環境中不可能配置專責人員 24 小時不斷的監控系統，因此整合於 Oracle Entrise Manager 中的 Oracle Diagnostics Pack 是最好的方案。

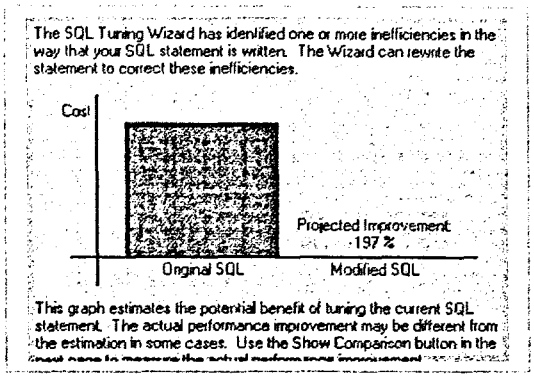
Oracle Tuning Pack

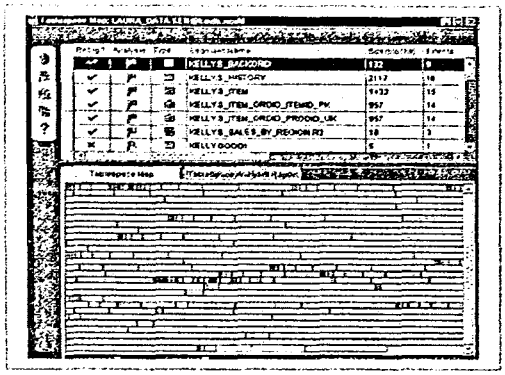
收集系統的數據只能了解目前系統的狀況，利用Oracle Tuning Pack可針對目前系統的狀況，自動產生分析報告並且提供改善效能的調校建議。Oracle Tuning Pack適用的範圍包含：資料庫結構：藉由調整SGA、排序、I/O、平行處理相關的資料庫參數，與作業系統相關的參數，以避免資源競爭的問題。SQL 語法的調校：Oracle Tuning Pack中的SQL Analyzer會自動分析SQL的語法，將不適當的SQL指令加以調整，以提昇執行效能。使用空間管理：透過圖形呈現表格空間的使用情形，藉由提高空間使用的效率及減少I/O次數，以改善資料存取的效能。資料存取最佳化：提供建立索引的建議，以改善資料存取的效能。

Oracle Change Management Pack

系統上線後，難免會需要配合業務的發展做修改，Oracle Change Management Pack 能夠針對資料庫的綱要修改，產生影響分析，避免因為修改不完全所產生的錯誤。例如為了加強客戶管理，客

戶主檔的表格重新設計後，利用 Oracle Change Management Pack 可找出所有參考的客戶主檔表格的預儲程式(stored procedure)、觸發器(trigger)或其他的表格。另外 Oracle Change Management Pack 也可以比較正式作業系統、教育訓練系統、品保系統和開發系統間資料庫物件的差異，並且提供同步的功能。





Oracle Application Manager

Oracle 內建的應用程式管理系統，其定義與管理系統的畫面如下面的圖行所式. Oracle Application Manager 是一個內建型的系統管理及監控解決方案，可完整掌握整個 Oracle 執行環境，讓 Oracle 保持高可靠度和高效能。除了被動的收集數據給管理者外，也可自行設定監控條件，當數據超過零界點時，主動以電子郵件或呼叫器通知相關人員。目前最新版的 OAM 已經與 OEM 整合在一起. 因此管理主機系統，數據庫系統，和應用程式系統將可透過單一系統介面完成.

當錯誤的情況發生時，例外管理會根據你的定義型態來處理. 換句話說，正常程式處理流成會轉移到例外管理程序的 PL/SQL block or subprogram. 內定的例外管理程序會隱含式的被啓動.