

UNIVERSITY OF TECHNOLOGY, SYDNEY
FACULTY OF INFORMATION TECHNOLOGY

**A UNIFIED APPROACH TO
ENTERPRISE ARCHITECTURE
MODELLING**

By Gerald R. Khoury

A dissertation submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing Science.

2007

To my lovely wife Isabelle, who has provided me with unwavering support throughout my research program.

And to my dear Mother and Father, for encouraging a love of learning.

Acknowledgments

I would like to extend my sincere thanks and appreciation to my thesis supervisor, Associate Professor Simeon J. Simoff. I am very fortunate that he accepted me as a student. Simeon's keen sense of diplomacy, combined with a kind and good-humoured nature, allowed me the room to grow and express myself through my work. All the while, Simeon's pragmatism, keen analytical mind and vision provided the framework needed to ensure a sound research program.

I would also like to thank my co-supervisor, Dr Roger Jenkins, for his kind and generous support. Roger provided me with valuable input when it was most needed, while also providing me with an inspiring introduction to the 'classics'.

A special thanks to the ITD Team at UTS who, despite their hectic schedules, were kind enough to support this research program. In particular, I would like to thank the Director Peter James for his generous reception to the project, Peter Demou and Ian Waters for their close involvement in the development of the project artefacts, and all of the ITD managers for their important contributions to the development of the EA models. I would also like to thank the independent Enterprise Architects who contributed to this research: Trevor Christie-Taylor, Luke Vassallo, Som Adel and George Wanat.

Copyright © 2007 by Gerald R. Khoury. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than Gerald R. Khoury must be honoured. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee.

Overview of Contents

List of Figures	i
List of Tables	iii
Abbreviations	iv
Abstract	v
1 Introduction	1
2 Enterprise Architecture and Systems Modelling	17
3 Metaphor	56
4 Theoretical Principles for The Development of Unified EA Modelling Languages	76
5 The LEAN Ontology	89
6 Experimental Research Methodology	112
7 Experimental Studies	119
8 Conclusions and Future Research Directions	153
9 Bibliography	159
10 Appendix A – Project Summary for UTS EA Project	169
11 Appendix B – Final Project Report for UTS EA Project	174
12 Appendix C – Questionnaires	211
13 Appendix D – USDoS ITA	218
14 Appendix E – USDoS ITA LEAN Models	261
15 Appendix F – Designing and Re-Engineering Subsystems	278

Detailed Table of Contents

List of Figures	i
List of Tables	iii
Abbreviations	iv
Abstract	v
1 Introduction	1
1.1 Research Problem	4
1.2 Research Hypothesis	6
1.3 Justification and Significance of the Research	7
1.4 Research Methodology	9
1.5 Thesis Outline	10
1.6 Outcomes	12
1.7 Expected Benefits	12
1.8 Summary	15
2 Enterprise Architecture and Systems Modelling	17
2.1 The Need for Enterprise Architecture	17
2.2 Defining Enterprise Architecture	19
2.3 Defining a System	20
2.4 Defining a Model?	20
2.4.1 Requirements of Modelling Languages	21
2.4.2 Models as Abstractions	22
2.4.3 Views and Viewpoints	23
2.4.4 Visualising Models	26
2.5 A Review of EA Modelling Languages	28
2.5.1 Unified Modeling Language (UML)	30
2.5.2 ArchiMate Enterprise Architecture Language	31
2.5.3 Integrated Definition Languages (IDEF)	33
2.5.4 Unified Enterprise Modelling Language (UEML)	34
2.5.5 Conceptual Graphs	34
2.5.6 Ad-Hoc Modelling Languages	36
2.6 A Review of EA Approaches	36
2.6.1 Soft Systems Methodology (SSM)	39
2.6.2 The Zachman Framework	41
2.6.3 The Information Framework	45
2.6.4 The Open Group Architecture Framework (TOGAF)	45
2.6.5 ISO Reference Model of Open Distributed Processing (RM-ODP)	46
2.6.6 Purdue Enterprise Reference Architecture (PERA)	47
2.6.7 Computer Integrated Manufacturing - Open System Architecture (CIMOSA)	47
2.6.8 Generalized Enterprise Reference Architecture and Methodology (GERAM)	48
2.6.9 IEEE Standards	49

2.6.10 Model Driven Architecture (MDA)	49
2.7 The Problem with Enterprise Architecture	50
2.8 Summary	54
3 Metaphor	56
3.1 Introduction	56
3.2 Contemporary Views on Metaphor	56
3.2.1 What is Metaphor?	56
3.2.2 Concrete Metaphor	59
3.2.3 The Ghost in the Machine	64
3.2.4 Persuasive Metaphors	65
3.2.5 Enterprise Architecture Metaphors	66
3.2.6 Metaphor Hierarchies	68
3.3 The Dynamic Type Hierarchy Theory of Metaphor	70
3.3.1 Support for Way's Theory	73
3.4 Summary	75
4 Theoretical Principles for The Development of Unified EA Modelling Languages	76
4.1 Introduction	76
4.2 Models and Metaphors	77
4.3 Model Hierarchies	79
4.4 A Methodology for Developing Unified EA Modelling Languages	82
4.5 An Enterprise Metaphor	84
4.6 Summary	88
5 The LEAN Ontology	89
5.1 Introduction	89
5.2 The Ontology Development Methodology	91
5.3 The LEAN Ontology	93
5.3.1 Purpose and Scope of the Ontology	93
5.3.2 Ontology Construction	93
5.3.2.1 Formalising the Enterprise Metaphor Concepts	94
5.3.2.2 The LEAN Topology	97
5.3.2.3 The LEAN Syntax	98
5.3.3 Assumptions	106
5.3.4 Limitations	106
5.4 The LEAN Modelling Tool	107
5.5 Summary	109
6 Experimental Research Methodology	112
6.1 Introduction	112
6.2 Quantitative Research Methods	112
6.3 Qualitative Research Methods	113
6.3.1 Action Research	114
6.3.2 Case Study Research	114
6.3.3 Ethnographic Research	115
6.3.4 Grounded Theory Research	115
6.4 Chosen Methods	116

6.5 Summary	117
7 Experimental Studies	119
7.1 Study One: Modelling a Large Enterprise	119
7.1.1 Introduction	119
7.1.2 Research Approach	120
7.1.3 Project Outline	121
7.1.4 The Survey Questions	122
7.1.5 Results	124
7.1.5.1 Business Users Survey Results	125
7.1.5.2 Enterprise Architect Survey Results	129
7.1.6 Analysis of Results	133
7.1.6.1 Closed Questions	134
7.1.6.2 Open Questions	135
7.1.6.3 Analysis of Models	138
7.1.7 Summary	142
7.2 Study Two: Re-Modelling a Public Domain Architecture	143
7.2.1 Introduction	143
7.2.2 The USDoS Enterprise Architecture	144
7.2.3 Analysis	146
7.2.4 Summary	150
7.3 Evaluation	150
8 Conclusions and Future Research Directions	153
8.1 Conclusions	153
8.2 Future Research Directions	155
8.3 Closing Remarks	157
9 Bibliography	159
10 Appendix A – Project Summary for UTS EA Project	169
11 Appendix B – Final Project Report for UTS EA Project	174
12 Appendix C – Questionnaires	211
13 Appendix D – USDoS ITA	218
14 Appendix E – USDoS ITA LEAN Models	261
15 Appendix F – Designing and Re-Engineering Subsystems	278
15.1 Developing a Unified Modelling Language	279
15.2 Reengineering Existing Interfaces	281
15.2.1 Decomposition and Mapping	281
15.2.1.1 Hotmail Interface	281
15.2.1.2 Yahoo Interface	283
15.2.1.3 Lotus Notes Interface	285
15.2.2 Interface Recomposition	287
15.2.2.1 Hotmail Interface	287

15.2.2.2 Yahoo Interface	289
15.2.2.3 Lotus Notes Interface	290
15.2.3 The Result	291
15.3 Developing New Interfaces	292
15.3.1 Identifying the Required Functions	293
15.3.2 Mapping	293
15.3.3 Construction	294
15.4 Conclusion	294

LIST OF FIGURES

Figure 1- Research Methodology	10
Figure 2 - Thesis Structure and Flow	11
Figure 3 - The relationship between Level of Description and Level of Machine Orientation for a selection of Modelling Languages	15
Figure 4 - Levels of Abstraction, Structures and Models Based on a talk given by Luciano Floridi, Informational Realism, Proceedings Computing and Philosophy conf., Canberra, Nov.2003	23
Figure 5 – An Example of the Decomposition of an EA into Various Architectural Views	25
Figure 6 - ArchiMate Concepts (Instituut, 2004)	32
Figure 7 - Situated Components of an Enterprise Architecture	37
Figure 8 - The Zachman Framework	42
Figure 9 - Possible modelling languages with which to populate the Zachman framework. From (Noran, 2003)	44
Figure 10 - The Relationship between the Source and Target of a Metaphor	57
Figure 11 - Scope and Level of Metaphors (partially based on (Hammond and Allison, 1987))	69
Figure 12 - Metaphor Hierarchy from Elastic to Concrete	70
Figure 13 - The Relationship between Models, Metaphors and Concept Type Hierarchies	77
Figure 14 - Enterprise System Hierarchy with Enterprise as Global Supertype	80
Figure 15 - Enterprise System Hierarchy with New Enterprise Supertype	81
Figure 16 - The Applicability of LEAN at Various Levels of Abstraction	84
Figure 17 - Methodology for Developing and Applying a Unified Language	85
Figure 18 - LEAN in relation to EA views and domain specific models	87
Figure 19 – Views, Viewpoints and Architectural Areas of Concern	96
Figure 20 - The Graphical Representations of LEAN Nodes	99
Figure 21 - A LEAN Relationship	100
Figure 22 - A LEAN Universal and Non-Universal Type	104
Figure 23 - Primary Research Approaches	112
Figure 24 - Responses for Two Groups (Business Users and EA's) Compared by Question and Test Area.	134
Figure 25 - Top Level Generic Activity Model	148
Figure 26 - Hotmail Interface	281
Figure 27 - Hotmail Functions	282
Figure 28 - Decomposition and Mapping of Hotmail Functions	283
Figure 30 - Yahoo functions	284
Figure 29 - Yahoo Interface	284
Figure 31 – Decomposition and Mapping of Yahoo Functions	285
Figure 32 - Lotus Notes Interface	286
Figure 33 - Lotus Notes functions	286
Figure 34 - Lotus Notes "Tools" Drop Down Menu	286
Figure 35 - Decomposition and Mapping of Lotus Notes Functions	287
Figure 36 - Recomposed Hotmail Functions	288
Figure 37 - Recomposed Hotmail Functions - Expanded	288
Figure 38 - Recomposition of Hotmail Interface based on Unified Metaphor	288

Figure 39 - Recomposed Yahoo Functions	289
Figure 40 - Recomposed Yahoo Functions - Expanded	289
Figure 41 - Recomposition of Yahoo Interface based on Unified Metaphor	290
Figure 42 - Recomposed Lotus Notes Functions	290
Figure 43 - Recomposed Lotus Notes Functions - Expanded	291
Figure 44 - Recomposition of Lotus Notes Interface based on Unified Metaphor	292
Figure 45 - Mapping of New Email Interface Functions	293
Figure 46 - New Email Interface Functions	294

LIST OF TABLES

Table 1 - Research Questions and Qualities Corresponding to the Research Hypothesis.	7
Table 2 – Some Integrated Modelling Languages	30
Table 3 - IDEF Methods from (IDEF, 1992)	33
Table 4 – Some Well Known EA Frameworks, Methods and Standards	38
Table 5 - City Landscape Metaphor Mapping	68
Table 6 - Definitions of Agent, Resource, Rule and Action Concepts	96
Table 7 - Mapping Between a Generic Relationship Set and the Range of Possible Node Pairings	105
Table 8 - ITD Project Objectives	121
Table 9 - Study One Test Areas and Research Approaches	123
Table 10 - Results of LEAN Survey for Business Users - Closed Questions	126
Table 11 - Results of LEAN Survey for Business Users - Open Questions	128
Table 12 - Results of LEAN Survey for Enterprise Architects - Closed Questions	130
Table 13 - Results of LEAN Survey for Enterprise Architects - Open Questions	133
Table 14 - Top Ten Words not in Wordlist	136
Table 15 - Concordance List for High Frequency Words	138
Table 16 - LEAN Node Frequency Distribution	139
Table 17 – Relationship Frequencies in ITD EA Models	141
Table 18 - Relationship Frequencies in USDoS ITA	149

ABBREVIATIONS

CG	Conceptual Graph
CRM	Customer Relationship Management
DTH	Dynamic Type Hierarchy
EA	Enterprise Architecture or Enterprise Architect
EA's	Enterprise Architectures or Enterprise Architects
EM	Elastic Metaphor
ERM	Enterprise Resource Management
HCI	Human Computer Interaction
HTML	Hypertext Mark-up Language
ICT	Information and Communications Technology
IS	Information System
IT	Information Technology
ITA	Information Technology Architecture
ITD	Information Technology Department of UTS
LEAN	Lightweight Enterprise Architecture Notation
MCS	Minimal Common Supertype
ODBC	Open Data-Base Connectivity
UML	Unified Modeling Language
USDoS	United States, Department of State
UTS	University of Technology, Sydney
VE	Virtual Environment
VR	Virtual Reality
WIMP	Windows, Icons, Mice, Pull-down menus

ABSTRACT

As IT environments grow in complexity and diversity, their strategic management becomes a critical business issue. Enterprise architectures (EA's) provide support by ensuring that there is alignment between an enterprise's business objectives and the IT systems that it deploys to achieve these objectives. While EA is a relatively new discipline, it has already found widespread commercial application. It is likely that EA will receive even more focus as IT environments continue to grow in complexity and heterogeneity.

Despite this widespread acceptance of EA as a valuable IT discipline, there are several serious challenges that contemporary EA approaches are yet to overcome. These arise from the fact that currently, there is no *unified* EA modelling language that is also *easy to use*. A unified EA modelling language is one that is able to describe a wide range of IT domains using a single modelling notation. Without a unified, easy to use EA modelling language, it is impossible to create integrated models of the enterprise. Instead, a variety of modelling languages must be used to create an EA, leading to enterprise models that are inconsistent, incomplete and difficult to understand. The need to use multiple modelling languages also places a high cognitive load on modellers and excludes non-IT specialists from developing or using these models, even though such people may be the most important stakeholders in an EA program.

The research presented in this thesis tackles these problems by developing a metaphor-based approach to the construction of unified EA modelling languages. Contemporary approaches to the understanding of metaphor are surveyed, and it is noted that one way to understand metaphor is to view it as part of a dynamic type hierarchy. This understanding of metaphor is related to the development of enterprise models and it is shown that highly abstract metaphors can be used to provide conceptually unified models of a range of enterprises and their component structures.

This approach is operationalised as methodology that can be used to generate any number of unified EA modelling languages. This methodology is then applied to generate a new, unified EA modelling language called 'LEAN' (Lightweight Enterprise Architecture Notation).

LEAN is evaluated using a mixed-methods research approach. This evaluation demonstrates that LEAN *can* be used to model a wide range of domains and that it is easy to learn and simple to understand.

The application of the theoretical principles and methodology presented in this thesis can be expected to improve the understandability and consistency of EA's significantly. This, in turn, can be expected to deliver significant tangible business benefits through improved strategic change management that more closely aligns the delivery of IT services with business drivers.

The findings in this research also provide fertile ground for further research. This includes the development and comparative evaluation of alternative unified languages, further research into the use of the methodology presented to align architectures at various levels of abstraction, and the investigation of the applicability of this theoretical approach to other, non-IT disciplines.

1 INTRODUCTION

"To make knowledge work productive will be the great management task of this century, just as to make manual work productive was the great management task of the last century." (Drucker, 1969, Section 6.1)

This thesis deals with the challenge of modelling enterprise systems in order to produce an enterprise architecture (EA). An enterprise is defined as a "... unit of economic organization or activity; especially a business organization." (Miller and Berger, 2001, p.4)

Models are essential for the design and development of effective Information Technology (IT) systems (Fox and Gruninger, 1997). In fact, Fox and Gruninger state that "Over the last 30 years, the role of enterprise models in the design and operation of enterprises has reached the point that few organizations of significant size can operate without them." (1998) The reason that they have reached this point is that the scale and complexity of these IT environments has, and is continuing, to grow substantially. However, in order for these enterprise models to be effective, the techniques for creating these models must be efficient, and the models themselves must be easy to understand (Theuerkorn, 2005 p. 14) (Chalmeta et al., 2001).

In this thesis, we examine the following three questions:

- How do we create consistent, high-level models that span multiple heterogeneous systems?
- How do we design these models such that they optimally harness the cognitive strengths of the human mind?
- How do we ensure that there is consistency between these high-level models and models at lower levels of abstraction?

These questions echo the three major challenges of enterprise systems modelling.

The first challenge concerns modelling scope. While many systems modelling languages exist, only a few have the semantic power to describe a wide range of system domains. Furthermore, the complexity of these languages makes them unwieldy and impractical for use in commercial applications. As a result, there is currently no widely

accepted method for describing multiple IT architecture domains (e.g. application, information, business and infrastructure architectures) using a single language (Iacob et al., 2002). Instead, multiple languages must be used to describe an EA. However, the requirement to be proficient in multiple modelling languages is simply too onerous for most enterprise architects: “there are too many enterprise modelling (EM) languages to learn and to understand” (Vernadat, 2002). As a result, the techniques used for EA modelling vary from company to company, and informal, undefined modelling constructs are widely used. (Sowa, 2000, p. 191) In fact, enterprise architects often create their own individual languages using informal pictures. (Instituut, 2005b) (Clements, 1996) Whether enterprise architects use formal, but disparate modelling languages, or single but informal languages, the outcome is the same: enterprise models that are inconsistent and incomplete. Furthermore, it is difficult to provide tools for the visualisation and analysis of such architectures. (Instituut, 2005b)

The second challenge in enterprise modelling concerns that of cognition. In order to create *effective* models, the models must be understandable. Given that the information domains being modelled may be highly complex, this is a difficult problem. Understandability arises when the user is able to transform the given models into effective *mental* models of the subject. Improved mental models lead to a better understanding of the subject area. With this understanding comes the ability to change and manipulate these structures to achieve better systems performance.

The third challenge in enterprise modelling concerns level of detail, or *abstraction*. Effective EA models are, by necessity, created at a very high-level of abstraction. This is in order to show the alignment between high-level business objectives and the information systems that support these objectives. Unless there is a logical translation (consistency) between these high-level abstractions and the consequent system and subsystem level models it will be difficult to implement decisions made at the strategic levels in a consistent and complete manner. Disjunctions between high and low level designs result in a lack of integration between the architectures at various levels of abstraction. For example, while the enterprise model may be designed along the lines of, say, a ‘city landscape’ metaphor, a system within this enterprise may be built using a ‘customer relationship’ metaphor, while the interface to this system may be built using a ‘car dashboard’ metaphor.

While not the *primary* focus of this research, this thesis does extend the typical area of study of enterprise architecture (that of modelling multiple systems in multiple domains) to address the problem of modelling at multiple levels of abstraction. Most modelling techniques model the enterprise at a single level of abstraction (Mili et al., 2002). The inability to use a single modelling language to model at various levels of abstraction is a problem that is largely ignored in the research literature, although there is one promising attempt in (Denford et al., 2004). This is probably because it appears that developing a single enterprise language is, itself, a daunting enough challenge, and making it work at various levels of abstraction just complicates the problem further. However, it will be found that the same theoretical principles that solve the problem of high-level modelling *across* an enterprise also show potential for structuring systems at lower levels of abstraction. To show this, the methodology that is used to structure the high-level, EA models, is applied to a very low-level sub-system modelling problem: the development of email interfaces (refer Chapter 15). However, a full development of this area of research is outside the scope of this thesis.

It has been suggested that the problem of developing a unified enterprise modelling language is formidable, and perhaps, intractable. Efforts using some of the more obvious, brute strength approaches add weight to this view. However, in this thesis a novel cross-disciplinary approach is used to attack this problem. Philosophic, linguistic and cognitive theory has been used to produce an approach that is novel and unique.

The result of this research is a methodology that can be used to generate a range of unified EA modelling languages, and a specific language that supports the development of unified EA models. In fact, it will be shown that the language 'LEAN' (Lightweight Enterprise Architecture Notation) is an effective language for unified systems modelling¹. That is, LEAN is easy to learn and use, and it supports the creation of unified models that are easy to understand.

The science of enterprise systems modelling is still in its infancy (Grefen, 1997) and there is great potential for new knowledge and research in this area. It is the intention of this thesis to add, in a significant way, to the body of knowledge already developed

¹ In this thesis the terms 'language' and 'notation' are used interchangeably to mean "a system for communicating".

by scholars and theoreticians of modelling, and hopefully, to generate to further research along these lines.

1.1 Research Problem

Two main factors influence the way that EA's are currently modelled. The first factor is the requirement to use a heterogeneous set of modelling languages to model an EA. "Today, no single existing modelling language by itself is capable of modelling all necessary aspects of an enterprise." (Noran, 2003) Instead, enterprise modelling requires a wide variety of modelling languages. This places high demands on EA developers and users to understand a wide variety of modelling languages. It also leads to inconsistent semantics and a weak ontological foundation, which can result in modelling inconsistencies and omissions.

The second factor is that systems modelling is dominated by the idea of metaphor. System modellers, consciously or unconsciously, use metaphor to design systems at all levels of abstraction. Without metaphor, it may be impossible to understand the new concepts that are required to analyse or use these systems (Lakoff and Johnson, 1980) (Lakoff and Nunez, 2000). The problem is that the metaphors used are often inappropriate and inconsistent. There appears to be little science directed to the development and implementation of effective system metaphors. As metaphor is so crucial to the development of effective models, and as they are so poorly applied to system modelling, we see problems at all levels of systems modelling.

At the level of enterprise modelling, these factors have resulted in a situation where it is currently not possible to create an effective, unified model that spans an organisation's systems. According to Noran (2003), the reason that we cannot currently create integrated² models of the enterprise, is because there is no commonly agreed metamodel and ontology for producing these models. Instead, a variety of modelling languages must be used: modelling languages that are actually designed for specific, individual IT domains. "Every domain speaks its own language, draws its own models, and uses its own techniques and tools. Communication and decision making across domains is seriously impaired." (Instituut, 2005b) This dependence on *domain-specialised*

² In this thesis the terms 'integrated' and 'unified' are used interchangeably to describe languages that can be used to model a wide range of IT domains.

modelling languages means that any large enterprise model is likely to be inconsistent, incomplete and difficult to decipher. A cross-domain, *unified* EA modelling language is needed to address these limitations.

These problems however, are not, confined to high-level systems modelling. Similar problems are also encountered when modelling small-scale systems such as a user interface for a single program. In fact, the accepted wisdom, and one that is typically taught to IT students, is that metaphors *should* be identified and applied to the design of computer systems. While metaphors are used abundantly, and usually with conscious intent, as a basis for the design of these systems, there is usually little consistency between the metaphors used for different system components. As users switch from one system component to another, they are likely to encounter a range of system metaphors. This can lead to confusion and disorientation.

Another major limitation with contemporary approaches to enterprise modelling is that the models are created at one, highly abstract, level of detail. There is then a disjunction between these enterprise models and lower level, more detailed domain models. The metaphor used to create the enterprise models is not continued down into the system and subsystem models, and this leads to inconsistencies between these structures. While consistency is difficult to define and measure, "it is recognised to be a major determinant of learnability." (Payne and Green, 1986) and inconsistency can lead to significant difficulties in delivering business value from these systems (NHS Quality Improvement Scotland, 2006).

Doumeingts and Ducq *define* enterprise modelling as "the representation of a part or of a set of enterprise activities at a global and a detailed level ..." (Doumeingts and Ducq, 2001). Certainly, there *are* benefits to EA frameworks and models that describe not just the enterprise level, but span all levels from strategy to implementation. These benefits include (Mili et al., 2002):

- Better support for re-use of detailed designs and implementations.
- Enforcement of adherence to the architecture style by actually implementing it.

In fact, Mili et al sees the goal of enterprise frameworks as follows: “Enterprise frameworks offer a unified view to model and develop enterprise information systems at every level of the vertical decomposition from the system infrastructure to the final application through the enterprise’s business model.” It will be shown, however, that this goal has not yet been achieved, and *cannot* be achieved effectively, without the availability of an effective unified EA modelling language.

It has been observed that the majority of contemporary EA approaches have been developed informally (Vernadat, 2002). This suggests that a scientifically grounded, theoretically sound approach to EA modelling may offer improvements over these methods. The current situation is summed up well by the following quote: “To date, there is no standard language for describing architectures; they are often described in informal pictures that lack a well-defined meaning. This leads to misunderstandings, and makes it very difficult to provide tools for visualisation and analysis of these architectures.” (Jacob et al., 2002)

1.2 Research Hypothesis

The hypothesis that is being tested by this research is as follows:

It is possible to develop a human-centred modelling language for creating unified models that span heterogeneous domains of an enterprise architecture.

This hypothesis can be broken into two research questions:

1. Can we develop a modelling language for creating unified models that span heterogeneous domains of an enterprise architecture?
2. Can such a language be designed to be human-centred?

The first research question concerns the development of *unified models*. That is, the use of a single, coherent language, to produce models that span *heterogeneous domains*. Detailed arguments justifying the need for such unified models are provided in Section 2.7. Criteria such as *effectiveness* and *relevance* can be used to determine how successful we have been in creating unified EA models.

However, the answer to the first question is perhaps trivial without addressing the second question. Theoretically, it is possible to aggregate the lexicons used to describe

every IT domain into one super-language with the semantic richness to describe the entire EA space (putting aside the problem of resolving the various ontologies at this stage). However, this language would be extremely unwieldy and difficult, if not impossible, to use. The models produced with such a language would also be extremely difficult to understand and hence, they would be poor communication vehicles and they would be difficult to verify. Therefore, it is the second question that gives flavour to this research. In order to have any value, the language must not only be unified, but must also be easy for humans to use and understand.

Thus, to prove the hypothesis presented above, it must be possible to develop a language that is *human-centred*. By human-centred, we mean that the language must possess the observable qualities of *learnability* and *useability* (measured relative to other modelling languages), and must be highly *understandable*. No matter how easy it is to produce models, if they are not easily understandable to others, then their use as a means of communicating between humans is extremely limited.

The relationship between the hypothesis, research questions and the desired observable qualities is shown in Table 1.

Hypothesis:	<i>It is possible to develop a human-centred modelling language for creating unified models that span heterogeneous domains of an enterprise architecture.</i>	
Research Questions:	Is it unified?	Is it human centred?
Qualities:	Effectiveness Relevance	Learnability Useability Understandability

Table 1 - Research Questions and Qualities Corresponding to the Research Hypothesis.

1.3 Justification and Significance of the Research

An EA aims to show how different systems and parts of the organisation interact and work towards fulfilling the organisation's objectives. An effective EA promotes "broad access to information, efficient re-use of IT components and solutions, and effective

global management of IT support." (United States Department of State, 1999) Ultimately, the value of an EA lies in the support it provides for developing effective IT systems. However, this can only be achieved if the EA models provide a coherent, consistent and complete view of the enterprise. "Because a coherent and integrated product is the ultimate goal, the models chosen must also be designed to integrate with each other." (Maier and Rechtin, 2000) This is not possible with today's modelling technologies.

Specialised languages have been developed for modelling *individual* architectural domains. However, none of these languages support the creation of high-level models that extend across multiple IT domains (Lankhorst, 2005 p. 83). Without such a language, there is no way to develop coherent and consistent EA models.

There have been a number of attempts to solve this problem and the solutions fall into two groups. In the first group are the integrated methods such as the Hatley-Pirbhai (H/P) and Quantitative QFD (Q2FD) methods (Maier and Rechtin, 2000, p.216) (Gruninger and Fox, 1996). The difficulty with these methods is that they are industry specific and only cover a small part of the EA domain. In the second group, we have symbolic logic notations such as predicate calculus and conceptual graphs. These notations are extremely rich in semantic power since they are closely related to natural language. However, they are also very difficult to use and understand, and impractical for the description of an entire EA.

The consequence of this is that EA models are developed using multiple languages, or informal languages (refer Figure 9). The use of multiple languages puts a high cognitive load on modellers and makes the communication of models very difficult, especially to those who are not technical specialists. The use of informal languages makes these models semantically poor and inconsistent. In either case, the EA models that are produced today are less than optimal and can be vastly improved through the availability of a human-centred, unified EA language.

"Enterprise Architecture is today widely spread among organizations all over the world." (Institute for Enterprise Architecture Developments, 2004) "Effective use of enterprise architectures is a recognized hallmark of successful public and private

organizations.” (United States General Accounting Office, 2003) It is likely that EA’s will continue to grow in importance as IT environments grow in complexity. This makes the need for modelling languages that support the goals of EA even more compelling. In addition, as the profile of EA increases, those in charge of the direct lines of business will seek hands-on ownership of the Enterprise Architecture. EA’s are too important to leave to the IT specialists!

Just as businesses sought to take greater control over application design and development in the 1980’s, we are likely to see EA ownership devolve to those who are making operational and strategic decisions upon which the business will succeed or fail. This devolution of ownership will drive the way we design and develop EA’s: they will need to be designed with a business perspective, rather than a blinkered focus on technology. Since business planners and developers, as well as IT specialists, must leverage these EA models in order to achieve their corporate goals, models that can only be understood by IT specialists are of little use. "What is needed are simple models that are easily communicated and models that can be tweaked and discussed with the employees who participate in the processes beeing (sic) mapped." (Rostad, 2000)

1.4 Research Methodology

This research draws upon a number of interdisciplinary areas including cognition, linguistics and philosophy, as well as computer science and information technology.

As Figure 1 shows, the research methodology is composed of a number of distinct stages that are executed as part of an iterative process. For instance, a review of the literature may result in a redefinition of the problem, results of hypothesis testing may lead to the hypothesis being updated, and so on.

Validation of the theoretical principles developed in this thesis is done using a range of experimental research methods. It is believed that the appropriate combination of complementary research methodologies is more likely to lead to valid conclusions. The case of the use of complementary research methodologies is further described in Section 6.4.

Two formal studies are presented where the theory has been applied and evaluated. In the first study, the unified language that has been developed (LEAN) is used to create EA models as part of an EA project within a complex IT organisation. The value of these models is then evaluated by the project participants using a survey approach. A separate group of Enterprise Architects is also surveyed in order to provide an independent perspective on the project results.

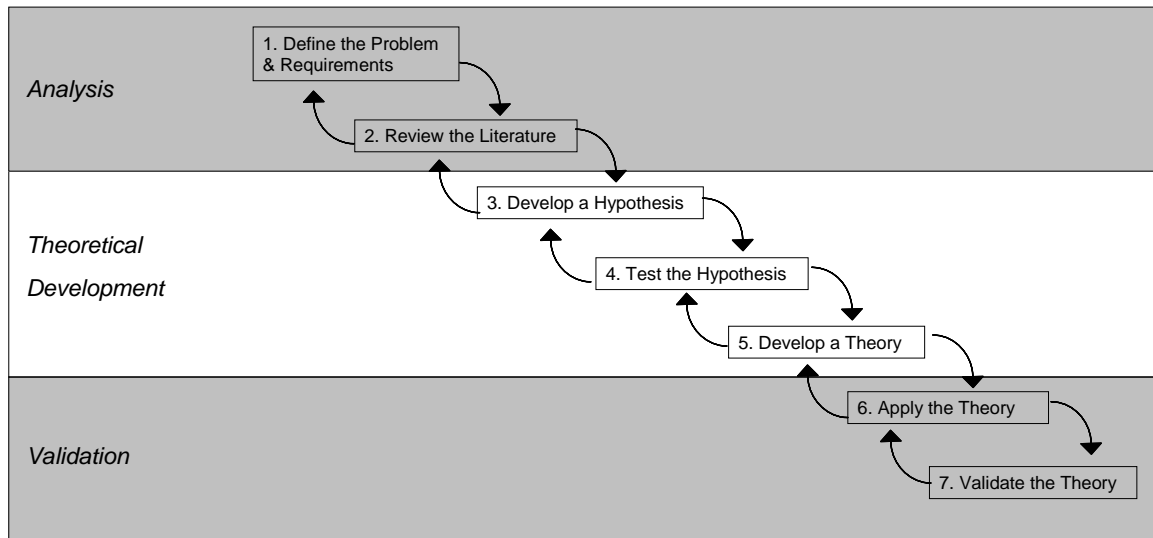


Figure 1- Research Methodology

In the second study, EA models and business scenarios from an unrelated, previously developed EA, are remodelled using LEAN and a comparative analysis is performed.

In addition, a third study is presented that extends the research provided presented in this thesis into a new area to show how a unified language may be used to provide structure alignment even at low levels of abstraction. This study is presented in Appendix F – Designing and Re-Engineering Subsystems.

1.5 Thesis Outline

The sections that make up this thesis and the logical flow between them are shown in Figure 2. Chapters 1 to 3 provide background material that will serve as a foundation for this research. Chapters 4 and 5 use this foundation to develop original theoretical principles for unified EA modelling. Chapters 6 to 9 present the research carried out to evaluate this theory. Finally, chapters 10 and 11 evaluate this research. A more detailed description of the individual chapters follows.

Chapter 1 provides a context for this research, describes its scope and explains why this research is important.

Chapter 2 provides a situated, contemporary view of EA and systems modelling. This provides an understanding of the role of EA modelling languages and the technologies with which it is used, such as EA frameworks, methodologies and standards.

Chapter 3 provides a contemporary understanding of metaphor in order to provide a theoretical foundation for the development of a new theory of enterprise modelling.

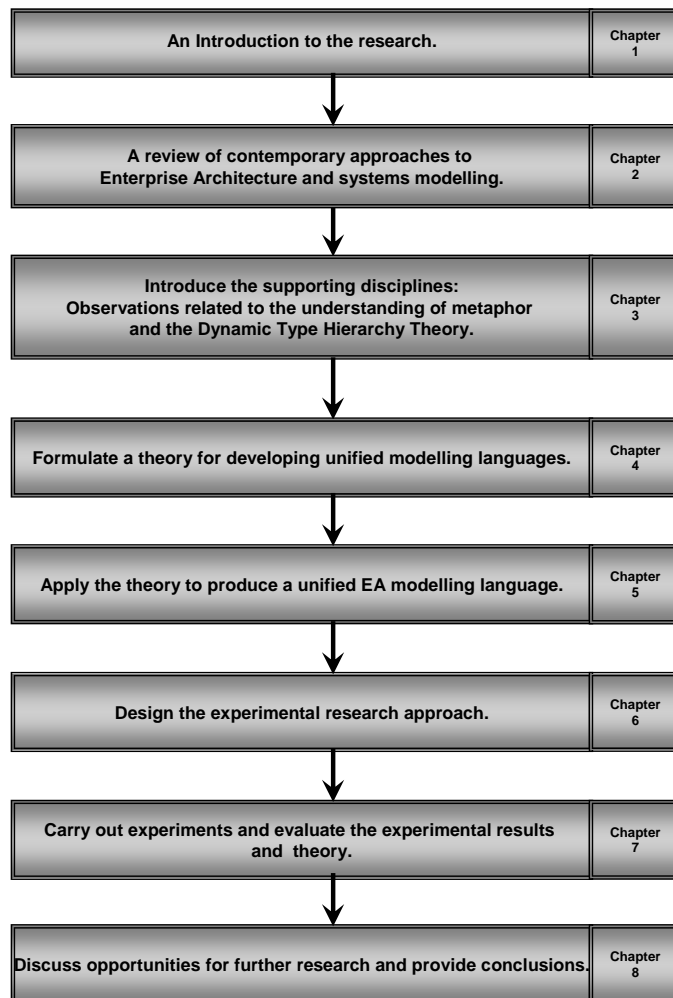


Figure 2 - Thesis Structure and Flow

Chapter 4 presents the original, theoretical basis for this research work.

Chapter 5 takes the theoretical principles used for developing unified EA modelling languages and applies it to develop one example of such a language. A candidate metaphor is developed as an ontology, which is then codified as a graphical modelling language.

Chapter 6 presents the research approach by which the developed language will be evaluated.

Chapters 7 presents two separate implementations of the language and provides an analysis of the results, an evaluation of the language and theory

Chapter 8 provides implications and opportunities for further research, and conclusions.

1.6 Outcomes

The outcomes of this research are:

- Theoretical principles and methodology for developing unified EA modelling languages.
- A fully specified, unified language for EA modelling (LEAN).

A number of published research papers have ensued from this research:

- Elastic Metaphors: Expanding the Philosophy of Interface Design (Khoury and Simoff, 2003)
- Enterprise Architecture Modelling Using Elastic Metaphors (Khoury and Simoff, 2004)
- Philosophical Foundations for a Unified Enterprise Modelling Language (Khoury and Simoff, 2005)
- Modelling Enterprise Architectures: An Approach Based on Linking Metaphors and Ontologies (Khoury et al., 2005)

1.7 Expected Benefits

As the importance of EA grows, in order to manage change within a global business environment that is increasingly dynamic and competitive, the need for more effective

approaches to EA modelling becomes paramount. Improved modelling of EA's will bring about considerable benefits arising from improved consistency and understandability. This will allow EA's to be used more effectively to manage change and ensure that IT solutions meet the needs of the business.

An EA that is modelled using a single language will have the following attributes:

- Greater explanatory power (models can be created that show the linkages and relationships between different domains and structures).
- Greater flexibility for the management of change and strategic planning (current disjunctions between systems are not 'hard-wired' into the model) (Veasey, 2001).
- Avoiding the loss of information that might occur in translating from one architectural view to another and ensuring cross-view consistency (Armour et al., 2003).
- Reducing cognitive load placed on a user that needs to understand a complex set of architectural views (Armour et al., 2003).

The use of a unified language to improve the integrity of conceptual EA models has consequences not only at an EA planning level, but also in terms of the systems that are subsequently developed. According to Brooks, "Conceptual integrity *is* central to product quality." (Frederick P. Brooks, 1995) In fact, Brooks believes that conceptual integrity is the most important consideration in systems design and the most important factor in the ease-of-use of a computer system. The use of an integrated modelling language that is based on a single metaphor has the potential to improve conceptual integrity.

Additionally, there are benefits to conceptual frameworks that describe not just the architectural level, but span multiple abstraction layers from strategy, down to implementation. These benefits include:

- Better support for re-use of detailed designs and implementations (Mili et al., 2002).

- Enforcement of adherence to the architecture style by actually implementing it (Mili et al., 2002).

As Figure 3 illustrates, there is a relationship between the level of description for which modelling languages are designed, and the level of machine orientation that they support. For instance, very coarsely grained languages such as natural language are very human centred. They are easily understood by humans because they make use of the cognitive abilities of humans and allow us develop useful cognitive maps. On the other hand, very finely grained languages are not natural for humans to use. However, these finely grained languages are much easier to translate into machine logic because of their formality and lack of ambiguity.

Much of the architectural work in recent years has been focused on highly formalised knowledge description using technologies such as XML and its derivatives (Murthy et al., 2005). This is because the benefits of autonomic computing can only be realised with the development of machine centred technologies that can precisely and unambiguously define the *meaning* of data. This approach ensures that EA components are formally defined; however, it does little to ensure that an EA is *complete* and *relevant*. To achieve this, we need technologies to describe high-level, coarse-grained features. By using the term “coarse grained” we mean that the language can be used to describe highly conceptual and abstract features. This is where languages such as LEAN are valuable. LEAN is designed to be human-centred and to describe relatively large, coarse-grained features. This ensures that the EA space has been adequately covered and provides high-level meaning to the whole EA endeavour.

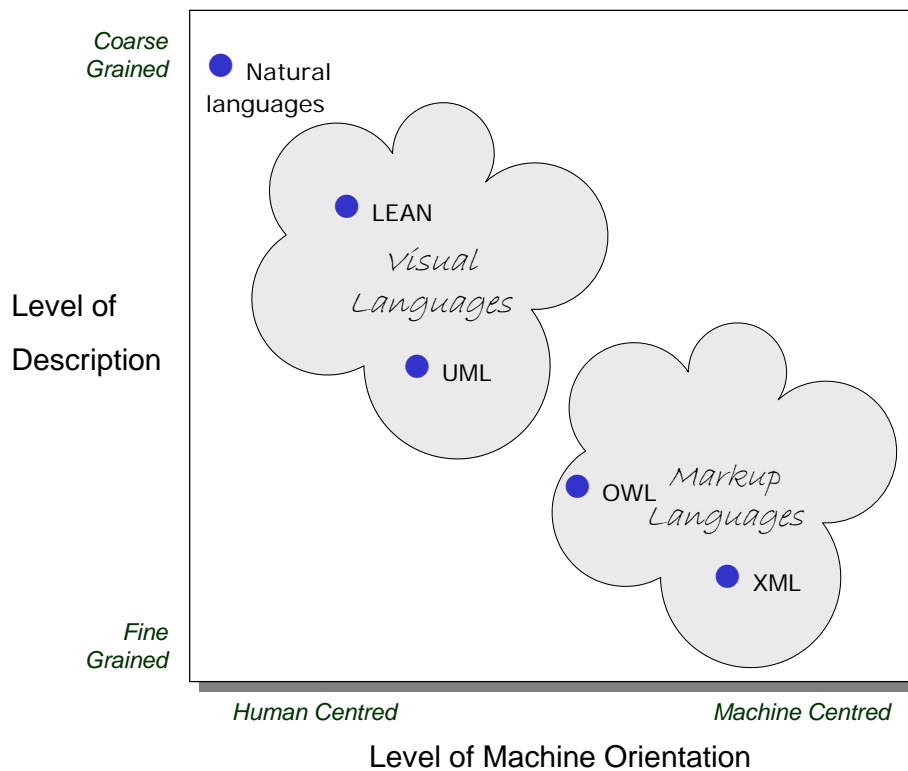


Figure 3 - The relationship between Level of Description and Level of Machine Orientation for a selection of Modelling Languages

1.8 Summary

This thesis deals with the challenge of modelling enterprise systems in order to produce an EA. In order to meet this challenge effectively, a unified EA modelling language is needed that can model multiple, heterogeneous systems across various EA domains. Furthermore, this unified language must be easy to understand by non-IT specialists as well as enterprise architects.

From a business perspective, a unified EA modelling language provides significant business benefits deriving from its ability to provide greater explanatory power and better support for strategic planning.

Currently, multiple modelling languages, or informal and undefined modelling constructs, are typically used to model EA's. This leads to inconsistent, incomplete and undecipherable enterprise models. These problems are compounded by the fact that multiple, inappropriate, or inconsistent metaphors are often used to describe the components of an EA.

This thesis takes a cross disciplinary approach to solve these problems. The result is a methodology for producing unified EA languages, and an example of one such language (LEAN). A range of complementary experimental methods is used to show that LEAN is not only unified, but also easy to learn, easy to use and easy to understand.

2 ENTERPRISE ARCHITECTURE AND SYSTEMS MODELLING

“All models are wrong. Some are useful.” George E.P. Box

This section provides a review of the literature on enterprise architecture and systems modelling. Following introductory discussions of EA, modelling and abstraction, the focus moves to systems modelling languages. This provides a basis for understanding EA's and how modelling is used to support the goals of EA. With this foundation in place, EA is formally defined and contemporary approaches to EA development are surveyed. Finally, the benefits of EA are surveyed, along with an analysis of the difficulties with current approaches to EA.

2.1 The Need for Enterprise Architecture

By 2007, 50% of Global 2000 enterprises will move beyond a pure technology architecture to include enterprise business architecture, enterprise information architecture, and enterprise solution architecture. By 2008, unified management and governed evolution of the enterprise architecture will become dominant best practices in 70% of Global 2000 enterprises. Meta Trends 2004/05

While EA is still in its infancy as a *research* area (demonstrated, for example, by the lack of doctoral research in this area) this has not prevented it from becoming an important, firmly established discipline within the IT industry. The reason for this can be understood by looking at today's economic environment, which is characterised by global competitive forces that are reshaping business dynamics. These forces include:

- Increased competition and rate of change arising from globalisation, deregulation and technology (especially information technologies).
- Increased reliance on information technologies to maintain a competitive advantage.
- Reduced product development times and accelerated life cycles.
- The decentralisation of computer resources.
- Exponential growth in the amount of information that needs to be managed.

These environmental pressures create new challenges that can only be solved by using new, more sophisticated approaches to the management of enterprise systems.

EA's are growing in importance as tools for managing change within this highly dynamic, demand driven, competitive business environment. As the rate of technological change increases, and as the information environment becomes more complex, more sophisticated methods are needed to manage that change effectively. EA's help to manage this change and overcome the problems of building isolated IT solutions that fail to support an enterprise's vision, goals and objectives. The lack of an architectural context can result in duplicated, poorly integrated, and costly systems. (United States General Accounting Office, 2003) "Understanding and visualising complex businesses enables you to identify and address areas that might be constraining business performance. Enterprise modelling helps you focus on those areas you can change, how these areas are currently functioning, how they might be optimised, and how any changes might impact other areas." (Fraser and Tate, 1995)

The potential benefits of EA's (a number of which are outlined in (United States Department of State, 1999)) include:

- Improved resource rationalisation.
- Cost reduction through improved efficiency and higher productivity.
- Improved data consistency and security.
- Improved planning capability and cost effectiveness.
- Higher flexibility and ability to respond to change.
- Improved synergies between systems, departments and companies.
- Better opportunity analysis, risk management and decision support.
- Faster development cycles.
- More effective integration and interoperability of systems.
- Improved encapsulation and preservation of knowledge.
- Improved access to corporate information.

"... to keep the business from disintegrating, the concept of information systems architecture is becoming less an option and more a necessity for establishing some order and control in the investment of information system resources." (Zachman, 1987)

“The benefits of enterprise architecting have begun to prove themselves: faster, better, and cheaper.” (Kaisler et al., 2005)

2.2 Defining Enterprise Architecture

There are myriad definitions of enterprise architecture (EA). Beznosov (1998) for instance, lists five different definitions for EA. Two more useful definitions are given in Kaisler et al (Kaisler et al., 2005). In general, the term ‘enterprise architecture’ is used when referring to architectures and concepts that encompass the whole of the organisation, including any or all of its processes, methods, assets and organisational intelligence. The EA provides a *comprehensive* view of these elements and their relationships. In particular, an EA is usually used to highlight the *alignment* between the business’ mission, goals and outcomes, with the provisioned IT applications, data and infrastructure that they rely upon.

As EA is a relatively new and evolving discipline, the term Enterprise Architecture is easily usurped to reflect the viewpoints and interests of differing user groups. For instance, an application programmer may see EA as referring to enterprise wide applications, while a business analyst may see EA as focussing on the linkages between an enterprise’s value chain and the supporting IT systems.

Within *this* research, the following EA definition is used:

An enterprise architecture is a holistic set of models that represent an enterprise, and its environment, in order to manage change.

In this context, the term ‘holistic’ refers to a set of complementary parts that are interdependent and where the focus is on the *whole*, rather than the individual parts. The observation that effective EA’s must provide a holistic set of models is acknowledged by other researchers (e.g. (Kaisler et al., 2005)). Yet, a review of the literature carried out for this research reveals no EA definitions that explicitly refer to this attribute. It has been included in the EA definition used in this research as it is considered to be the defining attribute that separates EA models from domain specific models.

Explicit in this definition is the notion that there will be interaction between an organisation and its environment. Organisations typically interact with the environment (other companies, customers, regulatory authorities, etc). In particular, the IT systems that support an organisation will have these linkages encoded, and the EA should be capable of modelling these interactions.

An important aspect of EA models is that they should be used to represent both the current and target architectures (Kaisler et al., 2005) (Gustas, 2005). An EA system must support the development of a 'roadmap' that shows how to progress towards the target architecture: an architecture that is aligned to the businesses strategy and goals. Thus, the definition given above explicitly refers to the role of EA's in managing change. In an environment that is not exposed to change, there is no need for an EA. Conversely, as the impact of change becomes greater, then so does the need for an EA to manage that change. Furthermore, a sophisticated EA can be used to leverage this change as a tool for gaining competitive advantage (Khoury, 2006a).

2.3 Defining a System

Since we will be making frequent references to enterprise information systems, it is important to define the concept of a system. Winograd and Flores define computer systems as "... collections of interacting components (both physical and computational) based on a formalization of some aspect of the world." (1987, p.83) This is useful; however, it neglects an important property of systems: that they are greater than the sum of their parts. In this research, we use the following definition of a system:

A system is a set of interacting components that exhibits emergent properties that are not exhibited by any of its individual parts.

It is possible then, to have a system of systems of systems. However, at each level, some new properties must emerge that are not exhibited by the sum of the individual parts, and that only emerge when all of those parts interact.

2.4 Defining a Model?

EA is concerned with the development of a holistic set of models. The word model is derived from the Latin word 'modulus', which is the diminutive of 'modus' which

means “measure” or “way of being”. Thus, ‘modulus’ means “small measure”, and model means “a smaller copy of the original”.

The concept of a model is concisely defined by Allen (1997): “Models are approximations to objects or processes which maintain some essential aspects of the original.” However, with particular respect to *systems* modelling, the following definition may be more suitable, as it shows *why* systems modelling is valuable: "Modeling is the creation of abstractions or representations of the system to predict and analyze performance, costs, schedules, and risks, and to provide guidelines for systems research, development, design, manufacture, and management." (Maier and Reichtin, 2000)

Models provide a tool for analysing complex systems environments and evaluating the possible impact of any changes (i.e. representing the temporal aspects of a system). They also facilitate communication by creating a “common frame of reference”. (Biemans et al., 2001) The technology of systems modelling can therefore be used to achieve common, and pragmatic, organisational objectives.

Models can take various forms. Black, for example, classifies models as scale, analogue, mathematical, theoretical, or archetypal. (1962) Later in this thesis, it will be seen that *metaphor* is also a type of model.

2.4.1 Requirements of Modelling Languages

One way of expressing models is through the use of a modelling language. Noran (2003) identifies two primary requirements of enterprise modelling languages that are required to produce meaningful models

- The language must be appropriate to the enterprise aspect being modelled.
- The language must be understandable by the target audience.

The target audience of enterprise modelling is diverse. While IT architecture specialists often take the lead in producing EA’s, they typically work in collaboration with non-architecture specialists and business people. The EA models are then used by a wide variety of people including business people who may have had little, or no, exposure to modelling or systems architecture. This makes the modelling of EA’s a challenging

problem and renders most current modelling technologies of limited use in communicating EA information to the audiences who may be most instrumental in influencing the enterprise's direction.

Noran (2003) also points out that “... *a balance must be struck between the expressive power and the complexity of the language.*” As the level of abstraction of a language decreases, the model resembles reality more closely. Thus, the model's expressive power increases. However, its complexity also increases, thus negating some of the benefit that accrues from a higher level of abstraction: the clarity that comes with the elimination of information that is not directly relevant to the analytical task at hand.

2.4.2 Models as Abstractions

“An abstraction of some system is a model of that system in which certain details are deliberately omitted.” (Smith and Smith, 1977) Abstraction allows us to simplify models (Wortmann et al., 2001) and allows analysts to focus upon specific characteristics of a system that are of particular concern to them. Those aspects are brought out more clearly, as all other aspects of the systems are hidden.

According to Biemans et al, “a model is - by definition - an abstraction of reality and therefore an incomplete representation of reality.” (2001, p.123) “The degree of abstraction and simplification depends on the interest of the targeted audience.” (Szegheo, 2000) That is, *all* models are abstractions: a system can be modelled *only* at a level of abstraction, and there is a relationship between abstraction and the concerns of the user; a topic that is discussed in more detail in the following section.

It is the quality of abstraction that makes architectural models so useful. Information overload is a particular problem in the analysis of IT systems, especially as they are becoming increasingly complex. It is only through the careful abstraction of the information that describes these systems, that we are able to perform useful analyses of this information.

Abstraction can also be useful in identifying fundamental components of systems that can be reused. Well-defined primitives (single variable models) can act as building blocks for developing composite models and systems. In this way, highly abstract

models are often used in systems architecture in order to determine a set of building blocks that can be used to create various new systems. These concepts are summarised in Figure 4.

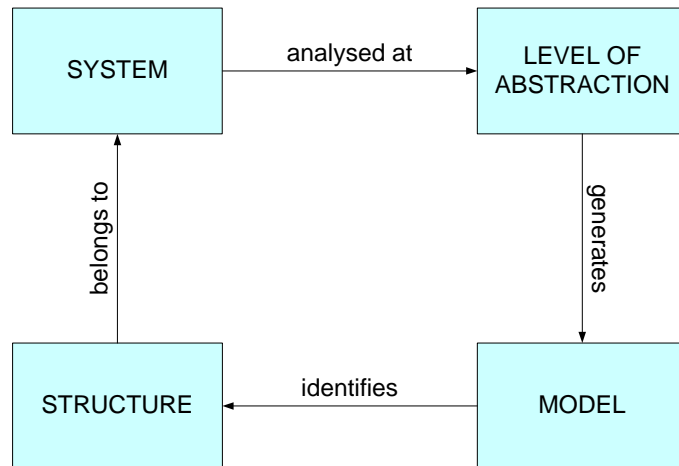


Figure 4 - Levels of Abstraction, Structures and Models

Based on a talk given by Luciano Floridi, Informational Realism, Proceedings Computing and Philosophy conf., Canberra, Nov.2003

2.4.3 Views and Viewpoints

EA's are used by a variety of stakeholders with widely varying backgrounds, interests, goals and responsibilities. An EA typically contains information that is of relevance to each of these stakeholders, but the threat of information overload means that it is ineffective to present all of this information to all users. Instead, the EA must present abstractions of this information. These abstractions are referred to as architectural 'views'.

A view is a collection of logically related models. (Maier and Rechtin, 2000, p.223)
More specifically, "A view is a representation of a system from the perspective of related concerns or issues." (IEEE 1471-2000)

A view generalizes the notion of a model, diagram, or other form of focused representation. Instead of attempting to say everything about an architecture in a single model, a view addresses a subset of the concerns for the whole system (architecture). This subset of concerns may be oriented toward a particular class of stakeholders (e.g., maintainers, thus a maintenance view) or toward specific system characteristics which may be of interest to several types of stakeholders (e.g., a reliability view for hardware suppliers, data designers, and software developers) or perhaps from other considerations or organizing principles.
(Hilliard, 1999b)

Views are thus closely related to the ‘separation of concerns’ principle (Dijkstra, 1976, Parnas, 1972). The separation of concerns principle says that complex systems should be decomposed so that different concerns or aspects of the problem are solved by different components of the overall system. This decomposition allows different parts of the problem to be solved in isolation, and then recombined to deliver the overall solution. This approach reduces the complexity of the problem down to a more manageable level (Figure 5).

The separation of concerns principle was originally derived within, and focussed on, a software engineering paradigm. However, it is just as relevant to the consideration of EA’s, where very large amounts of information need to be managed effectively to prevent cognitive overload by EA users. Views present an important mechanism by which a separation of concerns can be accommodated when presenting EA’s. Thus, “A view describes a system with respect to some set of attributes or concerns. The set of views chosen to describe a system is variable. A good set of views should be complete (cover all concerns of the architect's stakeholders) and mostly independent (capture different pieces of information).” (Maier and Rechtin, 2000, p.146)

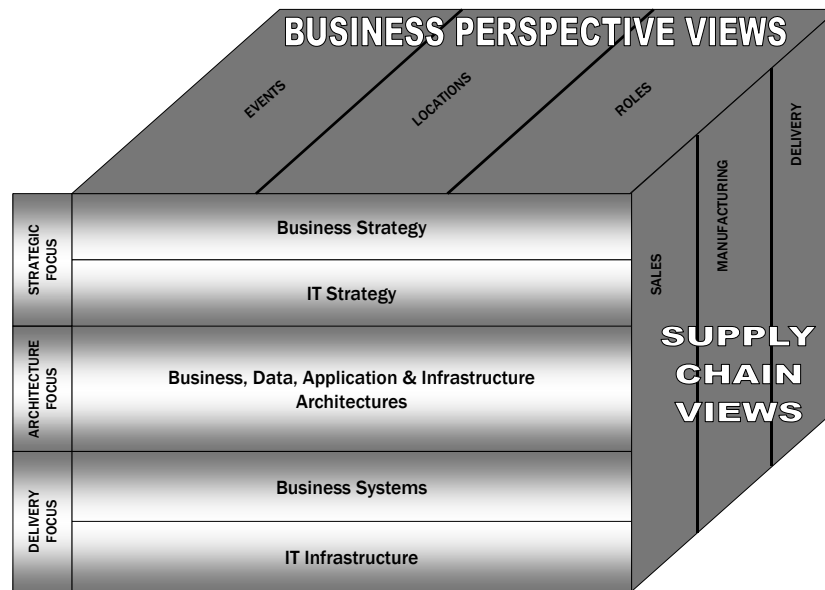


Figure 5 – An Example of the Decomposition of an EA into Various Architectural Views

Each type of EA stakeholder has different goals, interests and knowledge. The creation of views that support these different concerns facilitates the understanding and acceptance of the architecture by a variety of user and IT specialist groups and helps them to identify with it (Bernus and Nemes, 1994). For instance, “Customers may be satisfied with a ‘boxes-and-lines’ description; developers may want detailed component and connector models; managers may require a view of the corresponding development process.” (Medvidovic and Taylor, 1997)

The concept of views, is likely to grow in importance as enterprises “... improve their understanding of the different information needs of their users and customers at various touchpoints ...” (Gartner, 2002) In fact, Gartner make the prediction. “Through 2012, knowledge mapping³ (for example, through text categorization, indexing, and taxonomies) will be prevalent in nearly all information-rich applications (0.7 probability).” (2002)

While a view is a description of the system relative to a set of concerns, the actual set of concerns, a ‘viewpoint’, comprises the resources needed to address those concerns (Hilliard, 1999b). In the words of the IEEE, “A viewpoint captures the rules for

³ ‘Knowledge maps’ are a synonym for ‘views’.

constructing and analyzing a particular kind of view.” (2002) Each view corresponds to exactly one viewpoint; however, one viewpoint can generate many views (i.e. there is a one-to-many relationship between viewpoints and views).

One of the main difficulties with views arises from the fact that multiple languages are used to describe the various domains covered by an EA. Cross-domain views are fundamental for an effective EA. Yet, these languages typically require different methods of representation, making the creation of integrated, cross-domain views, highly problematic. The result is that the architect must be expert in each of the sub-domain modelling languages, and even then, they can only develop a very limited number of views. (Maier and Rechtin, 2000)

2.4.4 *Visualising Models*

If our extraordinary skill in perceiving the information inherent in the environment can be applied to data visualization, we will have gained a truly powerful tool. (Ware, 2000)

A number of graphical modelling languages have been developed that are designed to enhance the users’ ability to visualise the modelled systems. There is a strong imperative for this: the use of effective visualisation techniques can significantly improve the ability of users to assimilate and process complex information. “Visual representations ... play an important role in people’s ability to understand a problem and/or see a solution.” (Pawson, 2000)

Some of the advantages of visualisation techniques are described as follows (Ware, 2000) (Knight, 2002, , 2000):

- Visualisation provides the ability to reduce perceived complexity and increase the understanding of the user.
- Visualisation allows the perception of emergent properties that were not anticipated.
- Visualisation often enables problems with the data itself to become immediately apparent.
- Visualisation facilitates understanding of both large-scale and small-scale features of the data.

- Visualisation facilitates hypothesis formation.

“The eye and the visual cortex of the brain form a massively parallel processor that provides the highest-bandwidth channel into human cognitive centers.” (Ware, 2000) Consequently, the graphical presentation of information allows users to perceive and understand information *rapidly*.

It is clear that these qualities would be highly beneficial to the understanding and analysis of enterprise architectures. In particular, the ability to comprehend huge amounts of data is a problem that is particularly pertinent to the modelling of enterprise architectures, as they cover a number of domains across, what may be a very large enterprise. Arbitrary codes are hard to learn and easy to forget (Ware, 2000). For these reasons, the use of an EA graphical modelling language that leverages the capacity of humans to process information visually will be highly preferable to one that is non-graphical.

While much of the research on visualisation relates to 3D systems, most of the benefits associated with visualisation system can be obtained by using a 2D system. In fact, 3D visualisation systems are more expensive and difficult to navigate (Ip, 2001), without providing any significant advantages over 2D visualisation (Sutcliffe and Patel, 1996).

Since humans all develop essentially the same visual systems, regardless of cultural influences, it is believed that the same visual designs will be effective for all system users (Ware, 2000). This makes the development of effective visual modelling languages a more tractable problem.

One visual modelling approach is the use of ‘concept maps’. Concept maps are graphical representations that show the relationships between concepts. The relationships are represented as arcs, while the concepts are represented as nodes. “People find concept maps intuitive and easy to understand, and they are also amenable to formalization to provide computational services.” (Kremer and Gaines, 1996) “The nodes are labeled with descriptive text, representing the "concept", and the arcs are labeled (sometimes only implicitly) with a relationship type.” (Kremer and Gaines, 1996)

Concept maps have been used to create models for a wide variety of domains including education, management, science, social studies, politics and artificial intelligence (Kremer and Gaines, 1996) (Gaines and Shaw, 1995). Concept maps provide support for a wide variety of visual thought processes, and in particular, they can be used to support collaborative work (Gaines and Shaw, 1995). This makes the use of concept maps an interesting candidate for EA modelling, since the development of EA models is usually a collaborative and highly creative process.

2.5 A Review of EA Modelling Languages

Biemans et al categorise modelling languages related to business process design into several distinct “schools” (Biemans et al., 2001). These include:

- Object oriented languages based on Jacobson’s approach to software process design, such as UML.
- Human communications based languages such as the speech-act theory developed by Winograd and Flores. (Winograd and Flores, 1987)
- System dynamics methods that implement theories based on feedback loops.

Another way to categorise modelling languages is to segregate them into two groups: *domain specific* modelling languages and *integrated* modelling languages.

Domain specific modelling languages describe just one IT domain, e.g. computer applications, business processes or communications networks. Because of their specialised focus, they provide the expressive power to describe their domains in detail. The use of separate languages to describe these various domains mirrors the separation of concerns held by different stakeholders.

On the other hand, *integrated* modelling languages are designed to span *heterogeneous* IT domains. These languages are designed to model the entire, or a major part, of the EA domain in order to provide a unified, integrated view of the enterprise⁴.

There are a large number of IT modelling languages in use. However, “most languages are not really suitable to describe *architectures* ...” (Jonkers et al., 2003). Since the focus

⁴ Within this research, the terms ‘unified’ and ‘integrated’ are used interchangeably to refer to architectures that span multiple domains, or languages that can be used to model multiple domains.

of this research is *integrated* EA modelling, the literature review and analysis presented in this section is restricted to those languages that are acknowledged as having potential for modelling across multiple domains.

Individual languages can be seen as part of a continuum that extends from the universal, to the most specialised. On one end of the scale, we have the natural languages (e.g. English, in its various forms). These are the richest languages available, in terms of both breadth and subtlety. However, natural languages suffer from limitations in terms of formalism and ambiguity. At the other end of the continuum, we have languages such as formal mathematical languages. These languages are highly formalised, precise and unambiguous. However, they lack richness and subtlety. In order to develop an integrated modelling language, we must wrestle with this dichotomy. The question as to whether it is possible to develop a language that can span all systems and multiple levels of abstraction, while retaining a level of formality sufficient to allow precise systems specification, development and planning, is an open one that is partially addressed by this thesis.

It will be recalled that the potential value of integrated enterprise-wide models has been recognised for some time. Without integrated modelling, the benefits of an enterprise level view of the organisation are eroded. Thus, "Of special importance to architects are modeling methods that tie otherwise separate models into a consistent whole." (Maier and Rechtin, 2000) Yet, while there have been many attempts to develop these integrated models, there has been little success.

The remainder of this section investigates some of the attempts to achieve the goal of a unified systems modelling language. The goal is not to provide a detailed synopsis, or even introduction, to these languages. Most of these languages are well known and extensive descriptions and analysis are easily found in the usual literature sources. Rather, this review highlights and explains why the success of these languages as integrated EA modelling languages has been limited.

Table 2 lists the languages that are reviewed in this chapter. These languages were chosen on the basis that they are well known languages with some claim (not necessarily true) to being a unified EA modelling language, or that they are not well

known outside of academia, yet they provide some special characteristic which makes them important (e.g. Conceptual Graphs).

UML	Archimate
IDEF	UEML
Conceptual Graphs	Ad-hoc modelling

Table 2 – Some Integrated Modelling Languages

2.5.1 Unified Modeling Language (UML)

The Unified Modeling Language (UML) is sometimes *erroneously* viewed as a unified enterprise modelling language. In fact the term ‘Unified’ in UML, refers only to the fact that UML unified three previously distinct application modelling languages (Booch, OMT, and OOSE) (OMG, 2004, section 1.5). In fact, UML is not a single language, but a family of languages (Cook, 2005).

This does not rule it out as a suitable unified modelling language, as it is managed by a single body and the various elements are designed for compatibility. Also, UML is a mature modelling notation with wide acceptance and uptake within the software architecture community. UML is also widely supported by vendors and a wide variety of tools exist to support UML. Additionally, UML has been used with some success for modelling enterprises at high levels of abstraction. (Jackson and Webster, 2007) Nevertheless, the semantic integrity of UML models is not guaranteed, and there are a number of factors that make UML unsuitable for EA modelling.

According to OMG (the stewards of UML) “The Unified Modeling Language (UML) is a language for specifying, constructing, visualizing, and documenting the artefacts of a software-intensive system.” (OMG, 2004) While UML is designed for extensibility, it is not widely used outside of the software-development domain (Jacob et al., 2002). In fact, a number of researchers have found that it is not *suitable* for EA modelling (Boar, 1999, p.259) (Vernadat, 2002) (Gustas, 2005 p. 236) (Theuerkorn, 2005 pp. 14 & 16). It is designed primarily to represent single systems, and according to Armour (Armour et al., 2003) in its current form, without extension, UML cannot provide integrated representations across multiple systems, even within the software domain.

Even if UML were extended appropriately to cover all EA domains, the resulting complexity would likely make it extremely difficult to use, limiting its potential audience. In fact, even when constrained to modelling within the application domain, UML models have been shown to be difficult to comprehend by those who are not expert in its use (Arlow and Neustadt, 2003) (Lankhorst, 2005 p. 83). In particular, UML complexity makes it inaccessible to people such as managers and business specialists, who are important stakeholders of any EA project (Jacob et al., 2002) (Jonkers et al., 2003). In its current form, “UML is overly complex”: it is “part of the picture but isn’t the entire picture.” (Ambler, 2004)

2.5.2 ArchiMate Enterprise Architecture Language

The ArchiMate EA Language *has* been designed specifically as an integrated EA modelling language. The ArchiMate project is a product of a Dutch consortium between the Telematica Instituut and industry and academic partners. The goal of the ArchiMate project is to provide enterprise integration by developing an architecture language and visualisation techniques. A set of ArchiMate deliverables covering various aspects of ArchiMate and enterprise architecture is available from the ArchiMate website (Instituut, 2005a).

The ArchiMate enterprise architecture language “covers the business layer, application layer and technical infrastructure layer ...” (Instituut, 2005b). It is focussed on information intensive organisations and does not provide concepts to model physical products (e.g. physical stocks and logistics).

The ArchiMate metamodel consists of a set of architecture concepts. A graphical notation is proposed for representing these concepts, however ArchiMate state that this graphical notation could, in principle, be replaced by another notation. (Instituut, 2005b) Over fifty separate icons are provided in the ArchiMate notation to represent the concepts. These are shown in Figure 6.

As the ArchiMate concept icons present a comparatively large taxonomy, it is clear that a significant effort must be undertaken before gaining the skills to develop and understand a range of ArchiMate models. This precludes the use of Archimate by non-architectural experts. In addition, as the concepts are numerous, they represent the

enterprise at a relatively granular level of detail. While this has advantages for low-level systems planning and development it can act as an obstacle to high-level planning where there is a need to work at a high level of abstraction before moving on to lower level, detailed modelling.

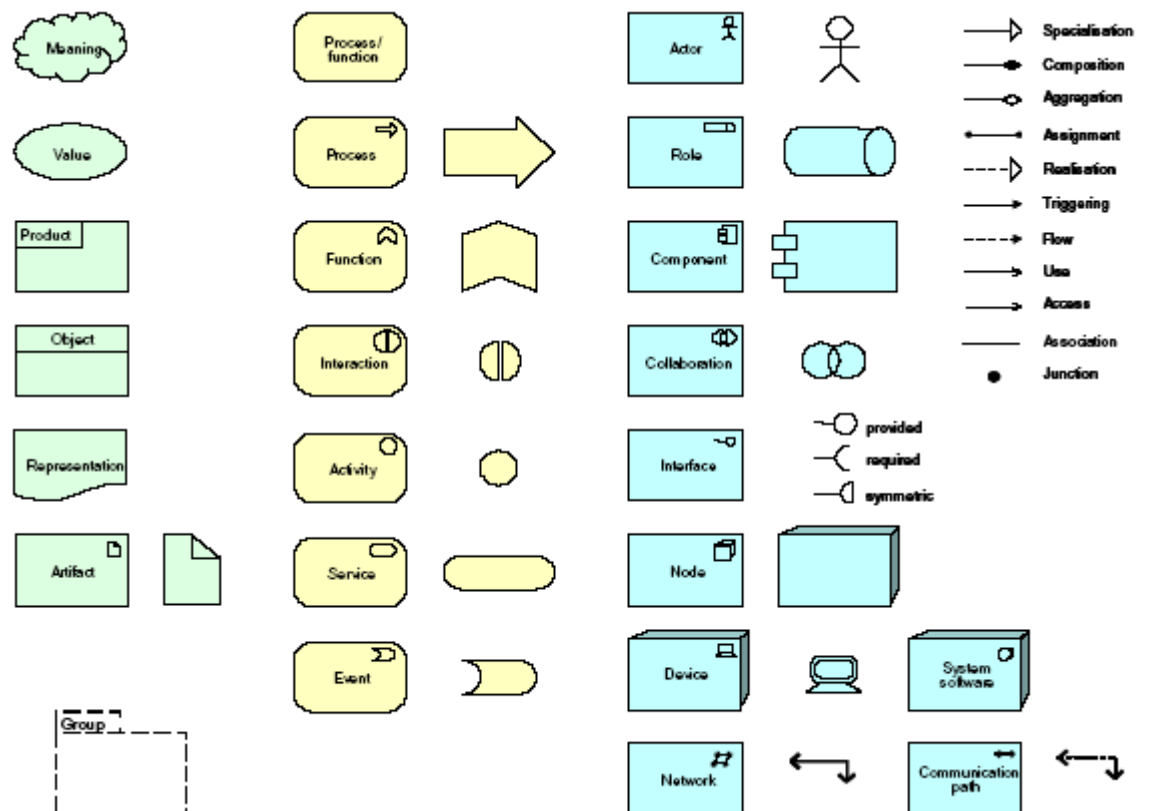


Figure 6 - ArchiMate Concepts (Instituut, 2004)

It should also be noted that the ArchiMate concepts are divided into a number of separate categories: Business Layer Concepts, Application Layer Concepts, Technical Infrastructure Layer Concepts and Relationship Concepts. Thus, while ArchiMate is designed to *integrate* across the domains of “organisational structure, business processes, information systems, and infrastructure ...” (Instituut, 2005b, p.v), in creating models that span these systems, the domain boundaries are not entirely dissolved. These arbitrary boundaries are still built into the models and form a basis upon which the ArchiMate language is structured.

2.5.3 Integrated Definition Languages (IDEF)

“The IDEF family of languages ... is the most widely used set of modelling techniques in North America.” (Vernadat, 2002) IDEF originated in the US military beginning in the 1970’s and was aimed at improving analysis and communications techniques (IDEF, 1993). There are currently sixteen IDEF methods, as listed in Table 3.

IDEF consists of a set of languages that can be used to create a structured approach to enterprise modelling and analysis (Knowledge Based Systems, 2004b). These include IDEF0 (or SADT), IDEF1x (or EXPRESS-G), IDEF3 and IDEF4. However, only three of these are widely used: IDEF0 (function modelling), IDEF1 (information modelling) and IDEF3 (process description) (Iacob et al., 2002).

IDEF Methods			
IDEF0	Function Modelling	IDEF7	Information System Auditing
IDEF1	Information Modelling	IDEF8	User Interface Modelling
IDEF1X	Data Modelling	IDEF9	Scenario Driven IS Design
IDEF2	Simulation Model Design	IDEF10	Implementation Architecture Modelling
IDEF3	Process Description Capture	IDEF11	Information Artefact modelling
IDEF4	Object Oriented Design	IDEF12	Organisation Modelling
IDEF5	Ontology Description Capture	IDEF13	Three Schema Mapping Design
IDEF6	Design Rationale Capture	IDEF14	Network Design

Table 3 - IDEF Methods from (IDEF, 1992)

IDEF0, IDEF1 and IDEF3 are all relatively simple, graphical languages. The IDEF0 function modelling language is designed to be capable of expressing systems at any level of detail (IDEF, 1993). The IDEF1 information modelling language is primarily aimed at manufacturing enterprises that are focused on systems integration (IDEF, 1992). The IDEF3 process modelling language is designed to be used at various levels of abstraction and across various business system domains (Mayer et al., 1995).

Thus, while the individual IDEF notations provide flexibility and ease of use, in order to model a complete EA, a number of separate notations are required. This would place a significant cognitive load upon modellers for the development of such architectures, and upon other non-IT specialists who may have to interpret such models. In addition, while the IDEF notations provide support for a variety of separate views, these views are all isolated from one another. There are no communication mechanisms between the various models because there is no over-

arching modelling language, and switching between views is not possible (Iacob et al., 2002) (Lankhorst, 2005 p. 33).

2.5.4 Unified Enterprise Modelling Language (UEML)

The Unified Enterprise Modelling Language (UEML) is designed to be a standard meta-model and underlying ontology that can be used as a unified language for describing enterprise architectures. It is designed “to provide a language that is easy to learn and easy to use and which can be provided as a standard user interface on top of existing systems.” (Vernadat, 2002)

This project is still in early development and according to the project website (<http://www.rtd.computas.com/websolution/Default.asp?WebID=239>), development activities were conducted between March 1st 2002 and May 30th 2003. The project has been conducted by a consortium of researchers funded by the European Union. It is not clear if this project is still active.

At this stage, the lack of development of UEML makes it difficult to assess the potential of this language for unified EA modelling.

2.5.5 Conceptual Graphs

The most generally applicable language is everyday language. However, written or oral descriptions of complex systems do not serve well as analytical tools or cognitive aids.

Symbolic logic notations such as predicate calculus and derivatives such as existential graphs and conceptual graphs have a direct mapping to language. As a result, they can be applied to many domains. For instance, English or other natural language sentences can be translated into an equivalent predicate calculus, or conceptual graph notation.⁵ These notations provide the full power of first order logic and more rigorous definition than normal language. Therefore, these languages have the semantic power to serve as unifying languages covering the entire EA domain.

⁵ That is not to say that *all* of the information in a sentence is fully conveyed, especially if the sentence is not completely literal.

The advantage of the conceptual graph notation over predicate calculus is the ability to represent models as structured diagrams. It is generally accepted that "... human understanding is improved by visual representations." (Polovina, 1993, p.37), and so conceptual graphs are *assumed* to leverage the human cognitive strengths associated with visual processing. However, a search of the literature does not reveal any research supporting this assumption.

Working on the assumption that conceptual graphs *do* leverage the visual cognitive strengths of humans, this would appear to make conceptual graphs an ideal tool for the representation of unified EA models. Indeed, Zachman and Sowa (1992) suggest using conceptual graphs as a means of describing any of the models that make up the Zachman Framework, as well as creating "... the metalanguage for talking about how the different levels relate to one another."

However, while Zachman and Sowa first suggested the application of conceptual graphs to enterprise modelling more than a decade ago, there is scant evidence of the application of conceptual graphs to real world EA problems. The primary reason for this is probably that conceptual graphs are too complex for general use as an effective modelling tool. This is demonstrated by the work of Polovina (1993), who investigated the suitability of conceptual graphs as a way of modelling strategic management problems within an accounting domain. Polovina found that "despite their strong *prima facie* attractiveness ... the inherent complexity of conceptual graphs fundamentally undermined them as a viable tool, other than for very trivial problems well below the level needed to be viable for strategic management accountancy." Although Polovina was working in a different domain, there are many similarities between the modelling domain that he was testing and the modelling of EA's. In addition, the senior practicing accountants who were his experimental subjects appear to have similar analytical competence to enterprise architects.

Ultimately, the experimental findings, coupled with the demonstrated lack of interest in conceptual graphs by the enterprise architecture community, show that the conceptual graph notation does not possess the attributes that are required for a language to be successful as a unified EA modelling language.

2.5.6 Ad-Hoc Modelling Languages

In many corporate situations, enterprise architects develop their own individual "home grown" modelling languages. These are often not formalised or standardised, even *within* a given organisation. Sometimes they are created in response to a lack of flexibility of existing modelling approaches. In other cases, it may be that the available modelling terminologies and ontologies are unfamiliar to the enterprise. (Szegheo, 2000)

Without a modelling standard in place, these ad-hoc models can be very difficult and laborious to read and understand. Even more importantly, the semantic integrity of the enterprise models is jeopardised. It then becomes impossible to rely on these models as a basis for organisational planning and design.

2.6 A Review of EA Approaches

Enterprise Architectures are typically developed using some, or all, of the following components:

- Reference architectures (which may contain generic models/modules).
- A framework.
- A methodology⁶.
- Standards.
- Modelling tools and languages.

EA frameworks are given particular attention in the literature and it is important to note the difference between an EA framework and a methodology. An EA *methodology* provides a technique for capturing various aspects of a business and turning these into models, while *frameworks* provide a structure within which these models can systematically be placed (The Open Group, 2002). Put another way, "Architecture frameworks are standards for the description of architectures." (Maier and Reichtin, 2000, p.221)

⁶ The terms method and methodology are often used interchangeably and, since that is the norm within the EA community, it is also the approach taken in this thesis. In the research literature however, 'methodology' can more formally mean 'the study of methods', or it may be used to refer to a *set* of methods.

EA Frameworks are useful constructs simply for the reason that, despite their apparent differences, most enterprises actually have a lot in common. The identification and codification of these common structural elements eliminates the necessity to develop new EA's from scratch. Instead, a relevant framework can be adopted and adapted for use. Reference architectures take this logic one-step further by tailoring an EA framework to a specific industry, perhaps with some models already partially developed.

STRATEGIC FOCUS	Business Strategy			
	IT Strategy			
ARCHITECTURE FOCUS	Business Architecture	Data Architecture	Application Architecture	Infrastructure Architecture
DELIVERY FOCUS	Business Systems			
	IT Infrastructure			

Figure 7 - Situated Components of an Enterprise Architecture

In practice, EA's can usually be broken down into a number of component architectures. The following set of domain architectures are usually considered to be the fundamental components of any complete EA:

- A business architecture.
- A data or information architecture.
- An application architecture.
- An infrastructure or technology architecture.

Figure 7 shows how these components provide the 'glue' that connects an enterprise's business and IT strategies to the delivery of its business systems and infrastructure.

Some of the most popular EA methods, frameworks and standards are listed in Table 4. These are described in the following sections.

EA Methods / Processes	EA Frameworks	EA Standards
Soft Systems Methodology Section 2.6.1	Zachman Framework Section 2.6.2	ISO RM-ODP Section 2.6.5
	Information Framework Section 2.6.3	IEEE Standards Section 2.6.9
	TOGAF Section 2.6.4	Model Driven Architecture (MDA) Section 2.6.10
	PERA Section 2.6.6	
	CIMOSA Section 2.6.7	
	GERAM Section 2.6.8	

Table 4 – Some Well Known EA Frameworks, Methods and Standards

This review presents some of the most well known EA approaches. However, an exhaustive review would be almost impossible. In fact, the following observations that were made by the IFAC/IFIP Task Force on Architectures for Enterprise Integration in 1991, is probably just as valid today:

1. *There are a very large number of architectures or models already in the literature or developed as proprietary projects by many industrial groups.*
2. *None of these were complete as yet.*
3. *Most present many of the same concepts but by means of different graphical and mathematical methods.*
4. *The ancient parable of the group of blind Indian philosophers who attempted to describe an elephant after each had felt only different separate parts, certainly applies here – Each of the proposed architectures is describing the same subject but from widely varied, and very incomplete viewpoints. Thus, the descriptions appear to be very different.*

(Williams and Li, 1998)

Some EA methods, frameworks and standards (such as the Enterprise-Wide Information Management (EwIM), the Method/1 Approach, and the ISMAP

approach) have been deliberately excluded from this review because they do not appear to be very well developed or formalised. Other approaches have been excluded because they appear to have been developed and presented in preliminary research papers, never to appear again. Many of these approaches have originated from academic research, or research and development laboratories (Mili et al., 2002). Consequently, it is likely that a good number of these approaches have found little acceptance or adoption outside of these environments.

There are also several examples of integrated approaches in the manufacturing domain, including the H/P, Q²FD methods (Maier and Rechtin, 2000, p.216) (Gruninger and Fox, 1996). However, it is apparent that these integrated methods are highly restricted in terms of domain. As a result, they are unable to deliver effectively the benefits that would be forthcoming from a method that integrated well over the entire EA domain. Also, "even in these domains the models are not in very wide use ..." (Maier and Rechtin, 2000, p.219) Additionally, it must also be noted that many organisations have their own proprietary or semi-proprietary approaches to EA.

The goal of this review, however, is to provide an understanding of the typical approaches to EA, within which, any understanding of the efficacy of an integrated modelling language must be situated. On this basis, the sample reviewed here has been selected in a way that identifies trends, and similarities, between the various EA approaches.

2.6.1 Soft Systems Methodology (SSM)

Soft Systems Methodology (SSM) (Checkland, 1981) was developed by Professor Peter Checkland and has a relatively long history. In contrast with 'hard' methodologies that are designed to deal with well-defined, technologically oriented problems, SSM is designed to tackle problem situations that are socially oriented.

SSM is usually described as being comprised of the following seven distinct stages⁷:

1. Analyse the problem situation.

⁷ It should be noted, however, that Checkland opposed this categorisation of SSM and developed a more holistic version of the methodology in CHECKLAND, P. & SCOLES, J. (1990) *Soft Systems Methodology in Action*, New York, Wiley.

2. Diagram the systems situation using 'rich pictures'.
3. Agree on 'root definitions'.
4. Build conceptual models of the desired systems.
5. Compare the desired systems with the current situation.
6. Agree on changes for moving towards the desired systems.
7. Develop an action plan.

There are two modelling stages in the SSM methodology: the current 'as-is' systems situation (step 2), and the desired 'to-be' systems situation (step 4). Rich pictures are recommended for step 2, although other notations can be used. Rich pictures are intended to be informal and imprecise: "Rich pictures are artistic and individualistic expressions, and therefore not 'right' or 'wrong'." (Couprie et al., 2004)

While the conceptual models created in step 4 *may* be based on a formal modelling language, no particular language for developing these models is mandated. A popular convention for SSM conceptual models is to use a very simple notation consisting of bubbles (activities) that are joined by lines. It is important to note that this is essentially an activity diagram. Consequently, there are many domains and problem types that this notation could *not* be used to model.

Checkland believes that "The complexity of the universe is beyond expression in any possible notation" (Checkland, 1981) and this is reflected by the fact that no formal notations are required to use SSM. However, this ignores the need for formal languages and the problems that arise when only informal notations are used. Without formal notations "Each analyst or team will develop their own style of Rich Picture." (Couprie et al., 2004) leading to the same problems that are encountered when Enterprise Architects use individual notations to create high-level models.

Fortunately, enterprise modelling is a much tighter domain than universe modelling, and the similarities between enterprises are strong. While no language may be perfect, the need for formal languages that avoid ambiguity and misinterpretation is essential, in order that the implemented systems satisfy the requirements of the enterprise. Consequently, a unified enterprise modelling language that is easy to use by non-IT

specialists would likely strengthen the SSM methodology by ensuring that all problem domains are modelled unambiguously.

SSM does not describe how to *build* a system, and is not intended to do so. It is “a way of securing commitment and taking into account a variety of interests.” (Underwood, 1996) As a methodology for addressing soft problems, it has a strong following. Eventually though, these problems need to be turned into hard solutions, and if these solutions are to be implemented, the requirement for a unified enterprise modelling language remains desirable.

2.6.2 *The Zachman Framework*

Zachman created seminal works in the area of enterprise architecture (Zachman, 1987) and (Evernden, 1996). Today, the Zachman framework (originally termed ‘A framework for information systems architecture’ or ISA) still provides one of the most popular approaches for describing enterprise architectures. The Zachman Framework is portrayed in Figure 8.

In essence, the Zachman framework provides a classification system for models that describe some part of the enterprise at a given level of abstraction. The framework comprises a matrix that describes the enterprise based on the different roles that an agent can take. These generic roles are represented by the various rows of the framework, consisting of planner (scope), owner (enterprise model), designer (system model), builder (technology model) and subcontractor (detailed representations). These descriptions are analogous to ‘architectural views’. “Therefore, the ISA framework serves as a convenient classification scheme or “periodic table” for information entities.” (Zachman and Sowa, 1992)

The Zachman framework does not provide a language for developing the models that go to make up each cell of the framework: each cell is expected to be modelled using “different techniques and different graphic representations ...”. (Zachman and Sowa, 1992) Indeed, the Zachman Framework was deliberately defined to be independent of any particular methodology or tool (Vail III, 2002). However, *suggestions* are given as to the appropriate models to be used for various cells.

	Data (what?)	Function (how?)	Network (where?)	People (who?)	Time (when?)	Motivation (why?)
Scope	<i>list of things</i>	<i>list of processes</i>	<i>list of locations</i>	<i>list of organizations</i>	<i>list of events</i>	<i>list of business goals</i>
Enterprise Model	<i>semantic model</i>	<i>business process model</i>	<i>logistics network</i>	<i>workflow model</i>	<i>master schedule</i>	<i>business plan</i>
System Model	<i>logical data model</i>	<i>application architecture</i>	<i>distribution architecture</i>	<i>human interface architecture</i>	<i>processing schedule</i>	<i>business rule model</i>
Technology Model	<i>data design</i>	<i>system design</i>	<i>system architecture</i>	<i>presentation architecture</i>	<i>control structure</i>	<i>business rule design</i>
Detailed Representations	<i>data definition</i>	<i>program</i>	<i>network architecture</i>	<i>security & access architecture</i>	<i>timing definition</i>	<i>rule specification</i>
System	<i>data</i>	<i>function</i>	<i>network</i>	<i>organization</i>	<i>schedule</i>	<i>strategy</i>

Figure 8 - The Zachman Framework

In Zachman and Sowa (Zachman and Sowa, 1992) an attempt is made to formalise the framework and provide a natural language that can be used as a unifying ontology for all the models required by the framework. This is in acknowledgement that "if the nature of the dependency between cells could be understood and stored in the repository along with the cell models, it would constitute a very powerful capability for understanding the total impact of a change to any one of the models, if not a capability for managing the actual assimilation of the changes." (Zachman and Sowa, 1992) Sowa proposes the use of 'Conceptual Graphs', a form of symbolic logic, to achieve this goal.

While some development guidelines are provided, the Zachman framework does not describe a method for developing the overall framework. The Zachman framework is simply a taxonomy that describes the enterprise using the metaphor of classical architecture, i.e. the design and construction of buildings. However, despite not being a method, the Zachman framework does embody an awareness of the system development life-cycle, albeit, a non-customary one: "The Zachman framework takes a

somewhat original approach towards life cycle, presenting life cycle phases as perspectives of the various stakeholders involved in the enterprise engineering effort." (Noran, 2003)

The Zachman framework has several shortcomings, generally arising from the fact that the Zachman framework is purely a taxonomy that classifies only the most transparent characteristics of an enterprise. As Beznosov suggests, a deeper analysis of the laws and principles that govern the rules for segmenting an enterprise would allow us to *understand* and *explain*, rather than just observe, these rules, and to discover new rules. (1998) However, as it stands, the Zachman framework does not provide this understanding and this may also contribute to the following limitations of the Zachman framework.

Implementation of the Zachman Framework is an extensive and arduous undertaking. The large number of cells makes the practical applicability of the framework difficult (Lankhorst, 2005 p. 25). While some of the cells can be modelled using well understood and accepted modelling techniques this does not apply to every cell. In fact, the modelling of some cells remains an open research problem. In particular, well defined modelling languages for describing the technical infrastructure are almost non-existent (Iacob et al., 2002) (Boar, 1999).

In addition, the problem of how the views *inter-relate* has been largely ignored. Zachman and Sowa acknowledge that the relationships between the cells are more important than the content of the individual cells (Zachman and Sowa, 1992). Yet, because the population of all thirty-six of cells requires almost as many different modelling languages, the cell relationships are extremely difficult to model (Figure 9).

The symbolic logic based Conceptual Graph notation offered by Sowa to solve this problem (Zachman and Sowa, 1992) has not been adopted by the architecture community (Beznosov, 1998). "Such logic expressions ... are too elegant for systems planners to understand and too difficult for systems developers to apply." (Wang, 1999) (also, refer to Section 2.5.5). As a result, the focus tends to be on 'filling in the boxes', and the Zachman framework, when implemented, can easily become a 'list' of applications and systems with little sophistication in terms of understanding how these

relate to each other. It is the information that sits at the edges of the boxes and the connections *between* the boxes that really conveys the most important information about the enterprise. This is the information that can only be gleaned once an EA has been developed: indeed, EA's are usually developed in order to reveal these relationships. Yet the fundamental basis of the Zachman framework is to segregate the enterprise into isolated units.

	What	How	Where	Who	When	Why
Scope (Contextual)	RP / English	RP / English	RP / Map	RP / English	RP / English	RP / English
Enterprise Model (Conceptual)	ER(M), IDEF1, UML Class	IDEF0, IDEF3 UOB, UML Act, GRAI Nets	Graph	Org Chart, GRAI Grid	GANTT / PERT, IDEF3 OSTN, Timed Petri	Struct English
System Model (Logical)	ER, IDEF1x, UML Class	UML Use Case Data Flow Diag.	UML Component	GRAI Grid, UML Use Case	Data Flow, IDEF3 OSTN, Timed / Colored Petri	FOL, Struct English, Z
Technology Model (Physical)	Relational, UML Class	UML Class, Activity, Structure Chart	UML Deployment	UML Real Use Case, UI Design	UML Sequence, Collab, State, Statecharts	FOL, Struct English, Z
Components (Out of Context)	DB Schema	Programming language	URL, IP, TCP/IP	UI Programming language	Struct English	Rule spec. In Prg Lang
Functioning Enterprise	DDL (SQL)	Machine code (0/1)	Address, Comm language	User / Worker	English (Schedule)	English

Figure 9 - Possible modelling languages with which to populate the Zachman framework. From (Noran, 2003)

Rather than allowing the development of multiple EA views based on stakeholder concerns, the Zachman framework assumes that there are only six discrete viewpoints, ranging from planner to user. This reflects the concerns of an enterprise from the 1970's and 1980's which were focussed on large-scale, mainframe-centric applications. However, as enterprises evolve, so their concerns change. Contemporary concerns, for instance, include areas such as security, governance, object orientation, change management, service oriented architectures and privacy. However, the Zachman framework does not support these concerns and they do not fit anywhere into the prescriptive framework.

Despite its popularity, the Zachman framework is founded on anecdotal evidence, untested observation and the conviction of a strong metaphor. Ultimately, the Zachman framework "lacks scientific foundation" (Beznosov, 1998) and "The large number of views is an obstacle for the practical applicability of the framework." (Jacob et al., 2002).

2.6.3 The Information Framework

The Information Framework (Evernden, 1996) is largely a development and extension of the Zachman Framework. It reuses many of the components of the Zachman Framework with some additional constructs. It was developed based on experience working in the Financial Services industry. Instead of using a building plan metaphor, the Information Framework uses a city plan metaphor.

One serious criticism of the Zachman Framework is the difficulty in completing all thirty cells. Unfortunately, the Information Framework extends this to fifty cells!

It becomes apparent from the development of frameworks such as this, that the structuring of the framework, or decomposition of the enterprise, is quite arbitrary. While Evernden tries to convince that his structure was "the correct one" (Evernden, 1996), it is clear that one can never develop the ultimate framework because it depends so much on environmental concerns such as individual industry characteristics, desired granularity, company objectives etc. Indeed, it appears that the focus of these types of studies has been on the 'softer' problem of framework structure, rather than the more profound problem of model integration.

Evernden does highlight the need for an integrated enterprise modelling language and methodology, suggesting that it might be possible to "chain together" techniques from different methodologies in some manner.

2.6.4 The Open Group Architecture Framework (TOGAF)

TOGAF (The Open Group, 2002) consists of three main parts:

- An Architecture Development Method (ADM).
- A repository of architecture assets - models, patterns, architecture descriptions, etc.
- A reference architecture.

TOGAF takes a platform centric view towards EA: a legacy of its beginnings as a Technology (infrastructure) framework. TOGAF allows the use of alternative architectural taxonomies and graphics.

TOGAF refers to “a continuum of architectures, architectural building blocks, and architectural models, that are relevant to the task of constructing an enterprise-specific architecture” (The Open Group, 2002). The level of detail increases as one moves through this continuum. TOGAF defines a range of Views and Viewpoints and recommends the ANSI/IEEE Standard 1471-2000 “Recommended Practice for Architectural Description of Software-Intensive Systems” for creating these views.

2.6.5 ISO Reference Model of Open Distributed Processing (RM-ODP)

The main aim of RM-ODP (also known as ISO International Standard 10746) is to provide portability of applications across heterogeneous systems (Raymond, 2006). It is defined as a standard by the ISO/ITU and has been produced in four parts: (ITU, 1996) (ITU, 1995) (ITU, 1995) (ITU, 1997) The standard is designed to be highly flexible and “considers distributed systems spanning many organizations and technological boundaries.” (ISO, 1995, Section 6.1)

Although the focus of RM-ODP is ultimately on applications, it is considered here as part of this EA review because it “manages complexity through a “separation of concerns”, addressing specific problems from different points of view.” (Vallecillo, 2001) These include the enterprise, information, computational, engineering and technology viewpoints. “The set of viewpoints are chosen to be both simple and complete and covers all the domains of architectural design needed.” (ISO, 1995) Thus, RM-ODP is far more than just an application framework. In fact, “The scope of RM-ODP is larger than just architectural description. RM-ODP makes extensive normative statements about how systems should be described, but also goes on to specify functions they should provide and structuring rules to provide those functions.” (Maier and Rechtin, 2000, p.227)

The five RM-ODP views are intended to be represented using different notations. The notations to be used are not specified, although responsibilities of the languages for each of the views have been suggested (Beznosov, 1998). In practice, UML has been used to represent the enterprise, information, computational and engineering viewpoints, and Technology Mappings for the technology viewpoint (Iacob et al., 2002).

“The complete specification of any non-trivial distributed system involves a very large amount of information.” (ISO, 1995) RM-ODP relies on the use of sophisticated viewpoints to reduce complexity and manage these vast amounts of information. Unfortunately, however, RM-ODP remains complex and difficult to use. It is a complex standard, compared to common practice in IT, and this acts as a barrier to its adoption. (Maier and Rechtin, 2000, p232) “In fact, its complexity and high-level of abstraction has discouraged many people from effectively using it for specifying and building open distributed applications.” (Vallecillo)

2.6.6 Purdue Enterprise Reference Architecture (PERA)

PERA was developed at Purdue University between 1989 and 1994. It is defined by the IFAC/IFIP Task Force on Enterprise Integration as a complete Enterprise Reference Architecture.

PERA views the enterprise as consisting of three major components:

- production facilities (manufacturing equipment)
- people/organisation
- control and information systems (information architecture)

These components are depicted as columns that begin with Enterprise Definition at the top and end with Enterprise Dissolution at the bottom.

The PERA approach is clearly focused on industrial/production types of enterprises. It is unsuitable as a generic framework that can be applied to a variety of industries.

2.6.7 Computer Integrated Manufacturing - Open System Architecture (CIMOSA)

The Computer Integrated Manufacturing - Open System Architecture is a reference architecture that has been created for use within the manufacturing industry (Neaga and Harding, 2005). This is reflected by the fact that, while CIMOSA provides a mature set of modelling constructs, they use an event-driven, process-based language (Vernadat, 2002).

In this regard, CIMOSA is a typical reference architecture: it makes assumptions about the nature and structure of the businesses that are to be modelled. This approach

makes reference architectures good starting points that can speed up the development of an enterprise specific EA's. On the other hand, reference architectures lack the flexibility to be easily extended to enterprises outside of the industry for which they are designed.

Thus, CIMOSA, while useful as an enterprise wide integrating architecture for the manufacturing industry, it is inherently complex (Szegheo and Gastinger, 2000) and has little applicability to the modelling of EA across a range of industries.

2.6.8 Generalized Enterprise Reference Architecture and Methodology (GERAM)

“GERAM was developed by the IFAC/IFIP Task Force to illustrate that all “complete” enterprise reference architectures should map together and have comparable characteristics and capabilities.” (Williams and Li, 1998) The Generalized Enterprise Reference Architecture and Methodology (GERAM) is comprised of three components:

- Generic Enterprise Reference Architecture (GERA)
- Generic Enterprise Engineering Methodology (GEEM)
- Generic Enterprise Modelling Tools and Languages (GEMT&L)

GERAM combines and builds upon the PERA and CIMOSA frameworks and like those frameworks, it is very process oriented. In addition, like PERA and CIMOSA, GERAM is focused on enterprise *integration*. It must be remembered that these frameworks were developed at a time when one of the main challenges that enterprises were facing was the integration of enterprise applications such as Customer Relationship Management (CRM) and Enterprise Resource Management (ERM) systems. Today's concerns are much wider in scope and include, for example, transformation towards service oriented architectures (SOA), the management of highly distributed applications, and the delivery of services over multiple channels. "GERAM alone cannot be used to engineer an enterprise; however, it should be used to assess what is needed for a given enterprise integration task (or task type)." (Noran, 2003)

2.6.9 IEEE Standards

IEEE Std 1471-2000 is IEEE's Recommended Practice for Architectural Description. While this is often generally referred to as an 'architecture' standard, it is specifically designed for software intensive systems.

"Originally, it was envisioned that the standard would codify the notion of view and prescribe the use of particular views. In the end, consensus only developed around a framework of views and viewpoints and an organizing structure for architecture descriptions, but there was no prescription of any particular views. As a recommended practice it is assumed that community experience will eventually lead to greater detail within the standard." (Maier and Rechtin, 2000, p230)

This standard is notation independent and does not provide or recommend a modelling language for describing architectures (Hilliard, 2000) (Lankhorst, 2005 p. 22).

2.6.10 Model Driven Architecture (MDA)

Model Driven Architecture (MDA) (Frankel, 2003) is focussed on software development. However, as it is designed to address the demands of *enterprise* computing, it is appropriate to address it here. While MDA is presented as a methodology, it is arguable whether it really presents the characteristics of a methodology as it does not provide a systematic guide to architecture development. It is categorised here as a 'standard' but it presents more as an embodiment of a *philosophy* about systems design. MDA is still immature and its development is incomplete.

The Object Management Group (OMG) is the custodian of MDA. MDA is built upon other OMG standards including UML, MOF (Meta Object Facility), CWM (Common Warehouse Metamodel), XML and CORBA. MDA is a new technology that is still being standardised and is described as an evolutionary step, rather than a radical departure from contemporary software development techniques (Frankel, 2003, p.1).

MDA is built on the assumption that enterprise architectures can only be built using a variety of "distinct but coordinated" modelling languages that target various levels of abstraction and domains (Frankel, 2003, p.58). Consequently, MDA can use a variety

of languages for modelling depending on which system aspect is to be modelled (Frankel, 2003, p.155). However, this is theoretical, and in practice UML is typically viewed as being the language with which all MDA models should be developed (Iacob et al., 2002).

Furthermore, while the MDA approach appears to be highly flexible, it makes the development of a complete enterprise model a very challenging problem. Even Frankel, who dedicates his book on MDA as follows, “This book focuses on MDA in the context of enterprise systems.” (Frankel, 2003, p.xvii) acknowledges the difficulty in using MDA in practice to create a complete EA when he states, “This book does not define a complete model driven enterprise architecture ...” (Frankel, 2003, p.58) Ultimately, this problem will not be solved until a unified EA modelling language can be applied within the MDA framework.

2.7 The Problem with Enterprise Architecture

EA is still in its infancy (Baker and Janiszewski, 2006) (NIH, 2004). Consequently, there are a lack of benchmarks or standards against which *good* EA can be measured (Parizeau, 2002). However, it is clear that there are serious weaknesses and limitations of current EA approaches, and that these prevent the full potential of EA from being realised.

One area of weakness relates to the metaphors used to describe EA's. For instance, the 'blueprinting' metaphorical approach to EA represents it as a static end-point of some process. Yet, this mistakenly implies that an EA is a single-use model; a static depiction of the organisation's current architecture with little or no support for simulating future target states (Presley et al., 2001, p.157). It should be noted that the source of this metaphor, construction blueprints, describes just one fixed and permanent state: the final design of the building or structure. However, enterprises change shape every day, and so the target state is a moving one!

In fact, there are two main types of enterprise architectures identified in the literature, "... architectures that represent the structure of a system at a given point in time (snapshot) and *life cycle* architectures, which describe the possible phases and artefacts involved in the life of a system (such as conception, development, build, operation,

dissolution etc.)." (Noran, 2003) An effective EA must be capable of describing both of these by providing a *dynamic* model. This ensures not only that the model can be shared, reused and kept up to date in a fast moving environment, but also that it can support decision making on an ongoing basis through 'what-if analysis', 'scenario planning' or 'simulations'. The blueprint metaphor is congruent with the 'snapshot' type of architecture, but is incongruent with these other non-static types of architecture.

EA method descriptions generally exhibit an appreciation of the need for architecture views. Yet, they lack detail as to how these would be defined or developed (Armour et al., 2003). The definition and creation of views can be very difficult when multiple modelling languages are being used, hampering "... the "flow" and dependency from elements in one view to the elements in another view ..." (Armour et al., 2003) While the use of views adds a great deal of complexity to the creation and maintenance of EA's, accurate and consistent representations of multiple views are critical for organisations that are operating in a rapidly changing environment (Armour et al., 2003). The development of an agreed metamodel and ontology for EA models would make it possible to look at any EA model using any of the views defined in any existing modelling framework, but this has not yet been achieved (Noran, 2003).

Traditional EA models are often unwieldy and difficult to navigate and explore. The problem is exacerbated by the fact that the stakeholders to which architectures are presented have "varied backgrounds, and technical and non-technical skill sets and interests." (Armour et al., 2003) Typically, these stakeholders include the CIO, a wide range of business users, system users, IT developers, systems analysts and systems architects. Business users, while responsible for determining the requirements for systems, generally are not fluent in any formal modelling notations (Cyre, 1997) and current standards for enterprise modelling are not oriented towards the business user (Chen and Vernadat, 2004, p.252). In fact, Bemelman and Dennis's investigation of architecture from a users' point of view (cited in (Rostad, 2000, p.136)), concludes that "the inherent levels of complexity and detail in most of the current architectures will become a major impediment to acceptance of these architectures in the industry." Solberg supports this finding: "Enterprise models are useful only if they are used. They will be accepted by users as a tool if they are *simple to understand, easy to use, computer-*

supported, and if they provide a realistic image of the reality. This explains the failure of many approaches proposed in the past ..." (2000, p.184 (my italics)) In fact, arguments over "which model is right", "which notation is right", and "which paradigm is right" are relatively meaningless if the model cannot be understood by the stakeholders." (Kaisler et al., 2005)

One problem presented by EA modelling is the scope problem: "The range of phenomena addressed by enterprise modelling stretches multiple disciplines, and accordingly many modelling languages and practices are used. This places high demands on IT architects to understand a wide variety of modelling languages, leading to long, complex projects that are out of date by the time they are completed (Beznosov, 1998). The use of multiple modelling languages leads to inconsistent semantics and weak ontologies. The models thus produced, may be inconsistent and contradictory (Bernus, 2001) (Roussev and Rousseva, 2004) (Jonkers et al., 2003). In fact, most available EA frameworks do not specify *any* language for modelling: "In effect, modellers are still left to create their own individual language." (Dewhurst et al., 2002)

Another limitation to the languages that are used to describe EA's is their restriction to a narrow range of abstraction levels. In fact, the syntax of EA modelling languages "... often closely resembles a programming language, which does not match the abstraction level of a system architecture." (Armour et al., 2003) There are currently no commonly accepted and used modelling languages that support the development of EA models spanning wide-ranging levels of abstraction. Unfortunately, there is usually a trade-off between the ability of a language to describe a wide scope covering multiple systems and its ability to model various abstraction levels. If priority is given to providing a wide scope, then some sort of mapping must be provided between the different levels of abstraction (Biemans et al., 2001, p.125).

To add to the complexity, different architecture modelling methods are used depending on the project life-cycle phase (eg. requirements gathering, conceptual, logical or physical design) (Bernus, 2003). In addition, different modelling techniques are used to represent different architectural *views*. For example, creating views to show system behaviour, data, performance, form, management or purpose all require

different modelling techniques (Maier and Rechtin, 2000). The integration across these views can be particularly problematic because “While the views are chosen to be reasonably independent, there is extensive linkage among views.” (Maier and Rechtin, 2000, p.146) These fragmented representations also make EA’s difficult to maintain (Gustas, 2005).

One particular area of difficulty is reconciling the disjunction between hardware models that are typically performance centric and physics based, and the software models that are typically object oriented and data based. (Maier and Rechtin, 2000, p233) Maier and Rechtin believe that rather than trying to reconcile these, we may be better to leave the disciplinary modelling methods as they are and look instead for ways to "develop inter-view consistency checking techniques." Even if this is achievable, this approach does not address the other disadvantages associated with using diverse modelling techniques, such as the difficulty in mastering many techniques in order to develop enterprise architectures, or the cognitive burden in trying to understand enterprise architectures that are composed of many model types.

Maier and Rechtin suggest that this problem might be solved by working “up from the engineering disciplines to create more general notations.” (2000) However, this approach has yet to yield demonstrated success. Taking a very different approach to the problem, Bernus believes that the problem itself needs to be restated: “... new criteria need to be developed for being able to develop shareable enterprise models.” (Bernus, 2003)

Two general solutions to the problem of integrated EA modelling seem apparent. These are similar to the approaches suggested for standardising object oriented application development methodologies (Henderson-Sellers and Bulthuis, 1998). One approach would be to identify *all* of the modelling elements currently used to describe different aspects of the enterprise and merge them into a ‘superset’ enterprise modelling language. The obvious drawback with this approach is that the ensuing language would be so complex that it would be impractical to work with. Alternatively, just the essential elements of various domain-modelling languages could be identified and merged into a ‘core’ enterprise modelling language. The problem with this approach is that the identification of essential elements would be problematic: it would

be very difficult to get agreement on what could be left out. In addition, the subsequent language is likely to be useful only for 'vanilla' applications.

Irrespective of the arguments as to the possible solutions, the *problem* of integrated EA modelling does exist, there is currently no solution, and its impact is highly deleterious (Gustas, 2005). A solution to this problem would have immense value: it would enhance the numerous benefits that are already acknowledged as deriving from the use of EA's, and it would add weight to the business justification for using EA to support organisational change.

2.8 Summary

As this Chapter has shown, there are a number of problems that arise from the fact that there is no *effective* unified modelling language that can be used to describe an EA (Noran, 2003) (Jacob et al., 2002) (Jonkers et al., 2003).

While there have been several attempts to develop languages that *can* model the entire (or a major) part of the enterprise domain, it is apparent that these languages have not gained widespread popularity. Furthermore, while there exists a number of different methodologies for developing EA models, the majority of these integrated methods are highly restricted in terms of domain, and "even in these domains the models are not in very wide use ..." (Maier and Reichtin, 2000, p.219). As a result, they are unable to deliver effectively the benefits that would be forthcoming from a methodology that integrated well over the entire EA domain.

The existing EA modelling languages that are not restricted in terms of domain are too complex for practical application to real world problems. The most promising EA modelling language appears to be the ArchiMate modelling language, which presents a wide variety of concepts that appear to cover the spectrum of EA quite well. However, with more than fifty concepts in use, its audience of potential users is unlikely to spread beyond that of specialised IT architects, and its use by business planners and strategists would, inevitably, be highly limited.

Ultimately, the use of multiple EA languages that cannot be integrated prevents the achievement of the prime goal of an EA: the development of an integrated view of the

enterprise (Maier and Rehtin, 2000) (Kaisler et al., 2005) (Beznosov, 1998) (Jonkers et al., 2003). In particular, EA's that are built using component-based frameworks (such as the popular Zachman framework) are fundamentally flawed because they implicitly model the enterprise as a set of independent structures with discrete boundaries. This leads to an EA model that provides a fragmented view of the enterprise with poor explanatory power and little flexibility for future planning.

It should also be noted that almost all contemporary EA approaches have been developed informally (Lankhorst, 2005 p. 41): "no scientific research methodology has been applied to this area." (Beznosov, 1998) This suggests that a scientifically grounded, theoretically sound approach to EA development may offer improvements over contemporary EA methods and frameworks.

3 METAPHOR

Figurative application of words ... are for nothing else but to insinuate wrong ideas, move the passions, and thereby mislead the judgment, and so indeed are perfect cheats ... They are certainly, in all discourses that pretend to inform or instruct, wholly to be avoided.”
John Locke, An Essay Concerning Human Understanding, 1689, (Bk.III, ch.10, §34)

3.1 Introduction

This Chapter provides a review of some of the contemporary perspectives on metaphor. This will provide a theoretical foundation for the development of a new theory of enterprise modelling by picking out some of the salient points made by contemporary researchers in order to lay the foundation for the ideas presented later in this thesis.

Some wider surveys of this domain have already been made. In particular, Way (1991) performs a careful and in-depth comparative analysis of many of the diverse theories that purport to explain the operative mechanisms involved in metaphor. In addition, Neale and Carroll review a number of metaphor taxonomies that have been developed by various researchers (1997).

3.2 Contemporary Views on Metaphor

3.2.1 *What is Metaphor?*

According to Johnson, a metaphor is “any image that represents one thing as something else in order to explain it better ...”(1994). Lakoff and Johnson similarly define metaphor as: “... understanding and experiencing one kind of thing in terms of another.” (1980) This latter definition provides the basis for the analysis of metaphor within this research. Furthermore, the ontology used in this thesis parallels the description of metaphor in cognitive linguistics, where the terms source and target refer to the conceptual spaces connected by the metaphor (Lakoff and Johnson, 1980). Figure 10 shows how these terms apply to the computer ‘desktop’ metaphor. The structure of the source domain is projected onto the target domain in a way that is consistent with the inherent target domain structure (Lakoff, 1993). Elsewhere in the research literature, the target is variously referred to as the primary system or the topic, and the source is often called the secondary system or the vehicle.

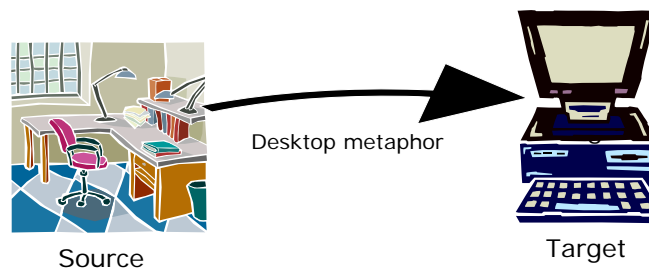


Figure 10 - The Relationship between the Source and Target of a Metaphor

Contemporary research has shown us that metaphor, far from being just a figure of speech, is actually central to everyday communication and learning (Lakoff and Johnson, 1980) (Lakoff and Nunez, 2000). In fact, it has been argued that only by relating a new concept (the ‘target’) to a well understood, everyday object (the ‘source’), can one develop new knowledge and understanding, and that without this process, it is *impossible* for humans to carry out abstract thinking (Indurkha, 1994). At a cognitive level, the source-target relationship is believed to produce a ‘conflation’ - a simultaneous activation of two different parts of the brain. New neural pathways are developed, and these synthesise the relationship as a single, entirely new experience (Lakoff and Nunez, 2000, p.42).

It is widely believed that metaphor is similarly valuable, and perhaps essential, to the design of effective *information systems*. In this case, metaphor is believed to provide value by:

- Reducing the effort required to understand the conceptual system model by providing an analogy between the new (target) system and a known (source) system.
- Assisting in specific task problem solving by allowing the user to extend their working knowledge of the target system, based on their understanding of the source system.

However, while humans use metaphor quite naturally and instinctively to learn and reason about new concepts, the explicit and overt application to IT domains *cannot* be assumed to carry the same benefits. In fact, Blackwell’s empirical research shows that metaphor has little effect on problem solving. “The generally assumed theoretical

benefits of user interface metaphor are supported by surprisingly little empirical evidence.” (Blackwell, 1998)

Indeed, much of the evidence that does support the use of metaphor appears to misattribute benefits from other sources onto the metaphor, or misattribute drawbacks of the metaphor onto other aspects of the system. For instance, the trend to incorporate metaphor into Graphical User Interfaces (GUIs), can easily lead one to misattribute the benefits of using a graphical interface to the metaphor that governs the choice of symbols and function (Blackwell, 1998).

Furthermore, there are examples of experiments where the use of a metaphor has been empirically proven to be misleading and confusing. Yet the metaphor is still declared successful on the basis that (a) the system probably wasn't being used correctly (b) the users *believed* the metaphor was valuable (Hammond and Allison, 1987). Similarly, Blackwell (1998) cites several other examples where claims are made purporting to show the value of metaphor in user interface design, *despite* evaluation results that show the contrary.

It may be that computing has presented society with such a 'radical novelty' and 'sharp discontinuity', that any use of metaphor and analogy to try to link new concepts to more familiar ones, is misguided: "...our past experience is no longer relevant; the analogies become too shallow; and, the metaphors become more misleading than illuminating." (Dijkstra et al., 1989) Perhaps we *should* throw away the metaphor, and "begin designing devices that have no metaphor, no real-world analogy." (Tristram, 2001) Yet, if everyday language is impossible without recourse to metaphor, how can we possibly hope to avoid using metaphor in a new field like IT? Is there a way to use metaphor that avoids its pitfalls while optimising its advantages?

These problems arise because, "the referents of computer metaphors are often ghostly abstractions, not things one can point to or see or touch." (Johnson, 1994) IT supports the creation of highly abstract environments, and yet, bound down in the garb of concrete metaphor, the opportunity to rise above the world of gross physicality is lost.

Still, metaphor remains an essential part of communicating and understanding. In fact, without metaphor, language and thought would be impossible. All knowledge is based

on metaphor (Indurkha, 1994); "... it is the basic means by which abstract thought is made possible." (Lakoff and Nunez, 2000, p.39) In fact, declarations such as those by Dijkstra that mathematics and formal logic are superior to metaphor are fundamentally flawed. Even mathematics uses metaphor freely (Lakoff and Nunez, 2000). Take for example, the concept of a function having a slope (Pimm, 1987) (Travers, 1996).

Johnson summarises the situation well, saying: "Having agreed with Kuhn, Black, and other philosophers of science that the use of metaphor is inevitable in human cognition, including scientific cognition, and having observed that it is used widely and with relish in computer science, I hasten to note that it is often enough the source of difficulty." (Johnson, 1994) This sets the scene from where we hope to move forward. We cannot escape the use of metaphor, and yet we still have much to learn in order to harness the power of metaphor for systems modelling.

The remainder of this chapter focuses on a several different types of metaphor and reviews the way in which they are applied. This investigation is situated within the context of enterprise systems modelling and is thus restricted in scope to those topics that lay a foundation for latter parts of this thesis.

3.2.2 Concrete Metaphor

Most contemporary computer system applications and interfaces are grounded in the world of metaphor. Most commonly, these are *concrete metaphors*. Concrete metaphors are based on objects that users are familiar with from their everyday experience (L'Abbate and Hemmje, 1998). For instance, we have the desktop metaphor composed of buttons, filing cabinets and trash bins, and windows through which we can view the world of information. And thus, "... we typically conceptualize the nonphysical in terms of the physical - that is, we conceptualize the less clearly delineated in terms of the more clearly delineated." (Lakoff and Johnson, 1980)

As we live in a physical world, it seems natural that computer interfaces should resemble as closely as possible – physical objects, especially objects that we have engineered and constructed to help us manage other aspects of the physical world. It seems logical, at first examination, that a metaphor that is based on well understood objects that we encounter and use every day should help us to understand the

unknown. After all, “The essence of metaphor is understanding and experiencing one kind of thing in terms of another.” (Lakoff and Johnson, 1980) This has been the prevailing school of thought when discussing the application of metaphor to computer design, and particularly Human Computer Interface (HCI) design.

However, there is another school of thought that the use of metaphor is *detrimental* to computer design. For example, Halasz and Moran state, “Analogy, *used as literary metaphor*, is effective for communicating complex concepts to novices. But analogy is dangerous when used for detailed reasoning about computer systems - this is much better done with abstract conceptual models.” (1982, p.386) Halasz and Moran believe that, inferences based upon an interface metaphor (metaphor based interface affordances) are likely to lead to invalid conclusions. The “information superhighway” metaphor provides a good example of this phenomenon. In reference to the information superhighway, Stefik states that “The metaphor has become so popular that it offers serious challenges to people talking about computer networks, because it carries with it misleading meanings associated with roads.” (1996) It has even been suggested that the tendency for metaphor to mislead is used as a *systematic deception*: “The animistic metaphor of the bug that maliciously sneaked in while the programmer was not looking is intellectually dishonest ...” (Dijkstra et al., 1989)

Sometimes, these problems arise because there is a misalignment between the metaphors used at different levels of system abstraction. For instance, a metaphor may be selected that it is believed will assist the user to understand an interface. Often, these metaphors are not related to structure of the actual program driving the interface (Ludewig, 2003). Under these circumstances, there are bound to be inconsistencies between the users’ expectations, and the functional behaviour of the system.

There may be a tendency to assume that the design of many engineered artefacts is close to optimal, since they have been used and refined over relatively long periods. For instance, we may assume that an office is the best way to organise a business work environment, a filing cabinet is the best way to organise information, a ‘clock face’ dial is the best way to display real variables, and so on. In a world defined by physical, engineering and economic dimensions, these assumptions *are* reasonable. However, in a computing environment these boundaries fade away. In this environment,

information storage, manipulation and representation are unencumbered by these limitations and there is potential to carry out tasks in a ways that mechanical devices just will not allow.

For example, the organisation of information may be modelled on traditional information management techniques that are based on physical manifestations of this information, such as books. However, as we digitise this information, new ways to manage and manipulate this information can emerge, and these new methods may break down many of our old assumptions. Concrete metaphors based on dual information structures (the Dewey decimal system for example, where there is a catalogue of sorted books) limit the potential usefulness of digital organising structures to those available within archaic physical limitations. Instead, a “third order of organisation” is needed to really leverage the power of information technologies, and this is based, in the case of information organisation, on “messy webs of information” that support serendipitous connections and innovative ideas (Weinberger, 2005).

Mechanical devices (a common source of concrete metaphor) are designed to *control* a single or small set of functions and/or to *display* a single or small set of variables. On the other hand, the information conveyed by a computer is often far more complex and diverse than would be conveyed by a single physical device. For example, in an aircraft, a single computer display could conceivably provide the control and display functions traditionally managed by hundreds of separate mechanical instruments. Therefore, there is a mismatch between concrete metaphor sources and targets. This leads to problems in three areas.

Firstly, the breadth of functionality required to replace myriad physical devices cannot be suitably conveyed using a single concrete metaphor. To overcome this limitation, convoluted aggregations of (often-unrelated) metaphors are created. These composites are intended to bridge the gap between what is available in the real world, and the extended ‘magical’ features that a computer can provide (Neale and Carroll, 1997). The composites form the modelling equivalent of ‘mixed metaphors’ and just as mixed metaphors are customarily prohibited in every day language (because they can lead to confusion and ambiguity), so there is a similar case for their exclusion from systems modelling.

When multiple metaphors are required to cover the target domain, it becomes difficult for the user to work out which metaphor applies to their particular problem, or to anticipate new metaphors they have not yet encountered. The development of a highly metaphorical computing language has led to the frequent juxtaposition of metaphor, leading to user confusion (Johnson, 1994). “Where metaphors do overlap, and where they could be interpreted as conveying contradictory information about the system, problems might occur.” (Hammond and Allison, 1987, p.83) It is the consistency and alignment with a thematic metaphor that promises to assist the user by enabling them to draw inferences about the system’s behaviour, and so “... where substantially new metaphors appear as the primary metaphor is unravelled, there is serious risk of confusion of thought.” (Black, 1979) In fact, Brooks believes that maintaining conceptual integrity is so crucial that it is better to *omit* system features and improvements, than to introduce features that do not integrate with the system’s basic design concepts (Frederick P. Brooks, 1995).

The second area of difficulty that arises from the use of concrete metaphor in systems modelling is that, since the depth of required functionality provided by concrete metaphors is rudimentary compared to the potential offered by computer systems, we find the metaphors twisted and *extended* in an unnatural manner. “After all the special addenda have been tacked onto the analogical model, the filing cabinet is no longer a familiar filing cabinet. Further, the addenda are the most important parts of the model.” (Halasz and Moran, 1982, p.384)

An example of this situation is the case of the city landscape metaphor, a concrete metaphor that is often used to describe EA’s. Dieberger, for instance, extends this metaphor and uses it not only to *describe* an EA, but as a basis for *developing* an EA system (Dieberger and Frank, 1998). Dieberger creates a system based on an “Information City” metaphor where “buildings act as containers for documents” and the façade of each building shows what type of information is contained within it (Dieberger, 1994) (Knight, 2002). This is justified on the premise that people “cope well with the complex task of reaching their working places or homes every day.”(Dieberger, 1994, p.10).

Of course, when we are travelling between work and home we are concerned with just two points, and so this presents an extremely low bandwidth situation that would be suitable only for modelling extremely small domains. On the other hand, our *information* needs usually span dozens, hundreds, or (especially with the Internet) millions of data sources and this data is constantly changing, reshaping and evolving. The use of a sequential search method where the user has to pretend they are walking along in front of buildings (Dieberger, 1994) is highly impractical. Given a choice of navigation techniques, one modelled on road transport is particularly limited.

In order to make this metaphor useful, Dieberger introduces “Additional magic features” (Dieberger, 1994) which bear no relation to the core metaphor. “Since cities are more a plane than a cube distances between documents can become unbearably long.” and so the cities become island cities that are “drifting ‘in the void’” and that can “expand dynamically ... similar to the stretching a (sic) rubber sheet.” These incongruent metaphors (incongruent with our understanding of how cities really are structured) are introduced in a desperate attempt to make the concrete metaphor workable. Thus, “Metaphor is extended and begins to take on the characteristics of imaginative narrative or myth.” (Johnson, 1994)

The third area of difficulty associated with concrete metaphors arises because features of the source domain can be projected onto the target domain, even though they are not an inherent part of the target. This is referred to as the metaphor over-attribution problem. This phenomenon leads users to attribute qualities to the target domain that may not exist, creating false expectations and reducing user performance.

At best, IT metaphors may be seen as perpetuating physical concepts that are no longer *necessary* in the IT environment (Pawson, 2000, p.61). However, in practice, the impact may be less benign. Gardiner and Christie sum up their view of the phenomenon as follows: “... by tying an interface to concepts which prevail in non-electronic environments, one is not taking full advantage of the benefits that can accrue from using the electronic medium. ... For example, a ‘filing cabinet’ metaphor can be as restrictive as the real-life filing cabinet.” (Gardiner and Christie, 1987, p.230) Similarly, Maher et al recognise that the use of spatial concrete metaphors such as rooms, although inspiring, may limit what can be achieved in the virtual world (Maher

et al., 2000). When information systems replace physical systems, there is no need to imitate the physical artefacts. In fact, building systems that imitate the existing world “... severely limits our possibilities.” (Ludewig, 2003)

3.2.3 *The Ghost in the Machine*

It has been observed that the majority of metaphors are anthropomorphic (Johnson, 1994) (Travers, 1996). In seeking to explain a complex domain for which an appropriate vocabulary has not yet been developed, we have drawn from another complex domain, one that is even more complex, yet exceedingly familiar – ourselves. Unfortunately, the consequences are not always favourable. In fact, it has been said that “the anthropomorphic metaphor ... is an enormous handicap for every computing community that has adopted it.” (Johnson, 1994) Others concur: “I have now encountered programs wanting things, knowing things, expecting things, believing things, etc., and each time that gave rise to avoidable confusions.” (Dijkstra et al., 1989) Anthropomorphic metaphors also carry a particular risk for over-attribution errors (Travers, 1996, p. 67).

Indeed, it is now apparent that computers are *rarely* viewed dispassionately, but are imbued with human characteristics including control, emotion and intelligence. Conversely, “... even the most objective of computer feedback can elicit psychological and emotional responses from users.” (Marakas et al., 2000) The ‘machine-being’ metaphor results in an over-attribution of characteristics from the source to target metaphors, leading to unrealistic expectations of computing technology (Marakas et al., 2000).

An interesting anthropomorphic metaphor is the ‘information agent’. Too often, ‘information agents’ conveniently take care of problems that we haven’t yet solved; a modern version of the ‘black-box where magic happens’! An example of this phenomenon is provided in the research reported by Thomas (1994). Because we know that human beings (the source of the agent metaphor) *are* intelligent and creative, devices such as these can easily deceive people into thinking that a problem has solved when it has merely been hidden behind the cloak of an anthropomorphic metaphor. Without recourse to such a metaphor, the lack of detail in this most critical part of the solution would be obvious.

3.2.4 *Persuasive Metaphors*

It is interesting to ponder the choices that have been made with respect to computing domains. While a superficial analysis gives us to believe that many of the metaphors in use are obvious choices, in that the mapping from source to target domains is relatively complete, further analysis shows that the selection of metaphor is not so arbitrary.

Take for instance the desktop metaphor. On one level, this appears to provide a fairly obvious mapping. Upon our desktops are a set of tools for carrying out business. We may want to use similar tools in the computing domain, and so there exists a fairly pragmatic relationship between the source and target. However, there are also many other source domains that provide similar mappings: a playground, a classroom, a garage, a library etc. Why are these metaphors not used?

On a deeper level, a metaphor does more than provide a set of attributes that map to the target. There is a larger meaning and connotation that we associated with each of these source domains. That is, the totality of each of these domains is greater than the sum of its components and upon deconstruction, the attributes that provide the *literal* meaning of the source metaphor fail to convey the whole meaning provided by the metaphor. As Black puts it, there is a “system of associated commonplaces” (Black, 1962) around each of these metaphors.

The desktop conveys more than just a work surface with commonplace tools. There are associations that might include concepts such as work, industry, ownership, enterprise, control, formality etc. Moreover, each of these may engender associated emotional responses that may vary from person to person. These associated commonplaces are likely to influence the user acceptance of a metaphor, as well as the cognitive and emotional responses to using that metaphor. Therefore, the choice of metaphor influences the direction, and degree, to which any computer system is *persuasive*. Because metaphor is inexorably woven into our way of thinking, it is routinely used, not just to communicate, but also to convince and persuade. Moreover, as we have seen, the success of metaphor as a persuasive device can lead researchers to confuse the persuasive strength of the metaphor with its success as a tool for learning and performance. In fact, in some cases the use of metaphor (for instance, a striking

visual metaphor), can serve as a seductive sales feature, particularly to a non technical audience (Pawson, 2000, p.61).

Once the effects of metaphor as a persuasive device as taken into consideration, the empirical evidence that metaphor provides HCI learnability or performance benefits diminishes. Still, the deliberate use of persuasive metaphor may be justified in order to influence the take-up and acceptance of computer systems; even if the benefits the user perceives are largely illusory. Some metaphors, are simply more pleasant to work with than others (Pawson, 2000, p.18). This may, for example, explain the allure of the ‘City Landscape’ EA metaphor.

A compelling and persuasive EA metaphor would need to have an emotional ‘hook’ that appeals to knowledge workers. The associations this metaphor would need to engender might include:

- control
- ease of use
- professional image
- responsiveness
- efficiency
- utility
- leading edge technology

The relevance of this understanding of the persuasiveness of metaphors will become apparent later in this thesis when a metaphor is *selected* as a basis for developing a unified modelling language.

3.2.5 Enterprise Architecture Metaphors

The term ‘Enterprise Architecture’ is a metaphor based on classical architecture - the design of physical structures such as buildings. This is heading towards becoming a dead metaphor as the term architecture is increasingly used in IT. However, when John Zachman (1987) first established the notion of information systems architecture, the metaphor was used very consciously. Zachman saw an analogy between the process of

classical architecture and the design of computer systems, and he projected the levels of representation produced by classical architecture onto the system development lifecycle.

This metaphor has been extended to cover the artefacts produced in carrying out EA activities. According to Presley et al, “An enterprise architecture can be thought of as a ‘blueprint’ or ‘picture’ that assists in the design of an enterprise.” (Presley et al., 2001) In fact, many well known approaches to EA are based around the 'blueprint' metaphor (for example, Boar’s book “Constructing Blueprints for Enterprise IT Architectures” (1999)). The process of EA development is seen as analogous to industrial design, and the established industrial design methods are seen as a rich source from which IT professionals can garner useful methods, techniques, and even vocabulary.

However, despite the popularity of the architecture metaphor there are significant and important differences between the process of systems design and the engineering design. In fact, systems design and engineering design are very different activities and require different process to achieve them successfully (Fowler, 2003).

A different, commonly used EA metaphor is the ‘city landscape’ or ‘city planning’ metaphor. For example, the United States Department of State – Information Technology Architecture publication describes architecture as follows: "An architecture is a guiding strategy or framework. ... It is not a detailed blueprint or wiring diagram, understandable only to technicians, but rather more like the city planning codes, zoning laws, and high-level plans that constrain the design, and enable the objective to be realized."⁸ (United States Department of State, 1999)

The city landscape metaphor is an example of a spatial metaphor (a special case of concrete metaphor). ‘Spatial metaphor’ is actually a label for a class of metaphors that include urban metaphors, landscape metaphors, geographic metaphors (that are navigated using latitude and longitude) and so on. “The appeal of the spatial metaphor is rooted in its simple and intuitive association with our experience in the physical world.” (Chen, 1999 p.178) Spatial metaphor is a popular approach to information

⁸ Regardless of this example, the blueprint and city planning metaphors are *not necessarily* mutually exclusive. In fact, they are often both used in describing EA’s where the city planning metaphor usually refers to the EA *process*, and the blueprint metaphor usually refers to the EA *products*.

space design and is the predominant metaphor in the world of virtual environments. “Spatial metaphor not only plays a predominant role in information visualisation, but also become one of the most fundamental design models of virtual environments.” (Chen, 1999 p.3)

The mapping between the source and target components of this metaphor is illustrated in Table 5.

SOURCE (City Planning) COMPONENTS	TARGET (EA) COMPONENTS
City vision and urban design.	IT Strategy and Planning activities.
Zoning and building codes.	IT Principles, standards and guidelines.
Maps and plans.	Architectural models.
Processes for changing city plans and allowing exceptions.	Architecture management process.

Table 5 - City Landscape Metaphor Mapping

3.2.6 Metaphor Hierarchies

While concrete metaphors are sourced on familiar objects, conceptual metaphors are sourced on structures of thought. However, these are not exclusive categories. Our understanding of concrete metaphors depends to some extent on our conceptualisation of the source object. Conversely, our understanding of concepts may be rooted in the physical (Lakoff, 1993). There is, in fact, a continuum that extends from the most highly concrete to the most conceptual metaphor.

There exists a relationship between concrete and conceptual metaphors with respect to “scope” and “level of description”. “Scope refers to the number of concepts ... that the metaphor addresses.” (Hammond and Allison, 1987, p.77) Level of description refers to the granularity of the knowledge being conveyed. Thus, *conceptual metaphors* are used to describe less granular knowledge structures, while *concrete metaphors* are used to describe highly granular knowledge structures. On the other hand, conceptual metaphors can address a wider range of concepts than can concrete metaphors. This is illustrated in Figure 11. Note that when these metaphors are used to describe systems,

they can overlap, and there can be gaps between them. When metaphors overlap, there is a contradiction where disparate metaphors are used to describe the same phenomenon. When there is a gap between the metaphors, there exists phenomenon with no metaphoric reference in use.

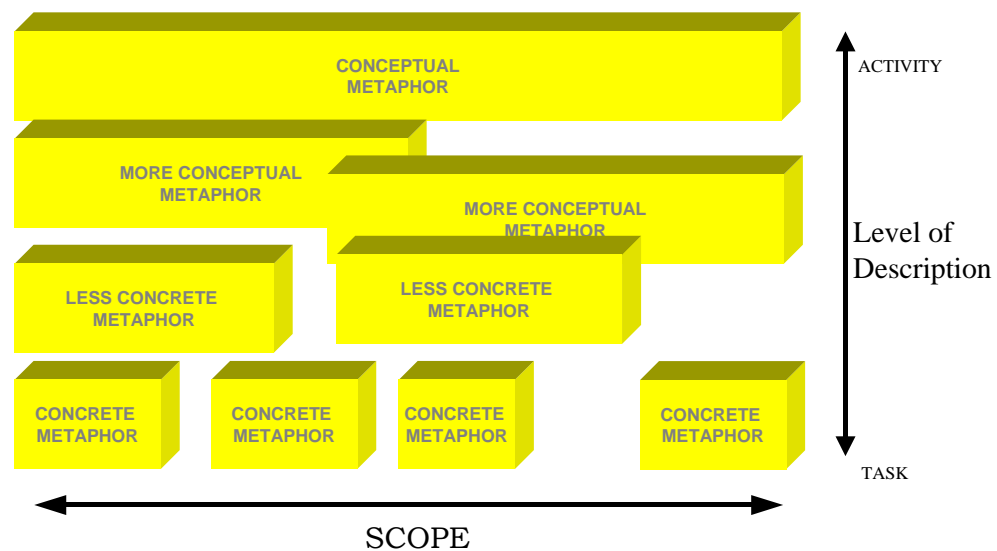


Figure 11 - Scope and Level of Metaphors (partially based on (Hammond and Allison, 1987))

Applying these concepts to the modelling of enterprise structures, it can be observed that larger scale structures are typically modelled using more 'conceptual' metaphors while smaller scale structures are modelled using more 'concrete' metaphors (Figure 12). It can also be observed that a conceptual metaphor is the least *leading*, and the least *misleading*, type of metaphor. Conversely, a concrete metaphor is the most leading, and the most misleading type of metaphor. For example, the concept of a learning organisation serves as a highly conceptual metaphor source. However, when applied to a specific target domain, this metaphor may actually say very little about that domain's function and structure. On the other hand, a filing cabinet is a highly concrete metaphor. When applied to a target domain it may provide some very specific information about that target's function and structure. It may also be very misleading because not all attributes of the source will map to the target and it may not be clear which attributes *do* map from the source to the target, and which do not.

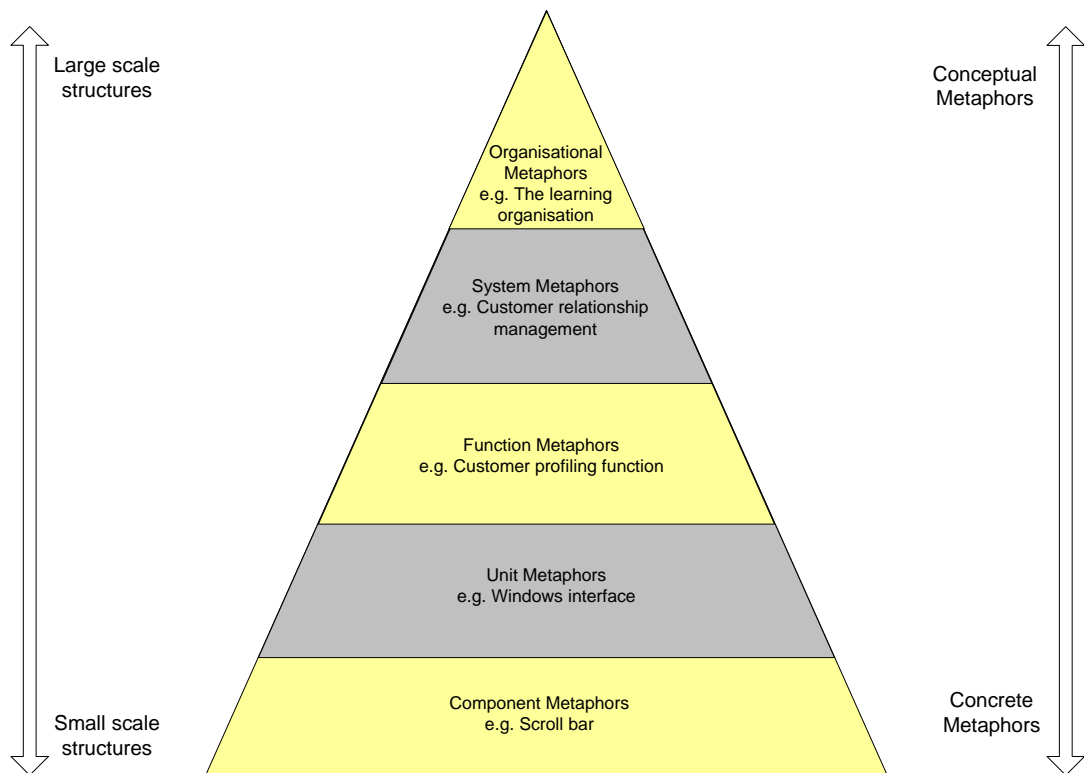


Figure 12 - Metaphor Hierarchy from Elastic to Concrete

3.3 The Dynamic Type Hierarchy Theory of Metaphor

In the previous section, we alluded to the hierarchical nature of metaphor. We will now build upon these concepts and adapt Eileen Way's dynamic type hierarchy theory (DTH) in order to formalise a theory for developing unified EA models.

The DTH theory “involves a theory of metaphor that incorporates Sowa's conceptual graphs, dynamic type hierarchies, and Max Black's interaction approach.” (Way, 1991, p.125)

Numerous theories have been developed explaining how metaphors are used and why they work. These theories include emotive theories, the substitution approach, the comparison theory, metaphor as analogy, the controversion theory and Chomsky's anomaly model. Way surveys and analyses these approaches and illustrates the shortcomings and contradictions that beset them. Way finds that the interaction view, while not perfect, is the most promising contemporary theory of metaphor and that “much of the experimental data of metaphor is either compatible with or actually supports aspects of the interaction view.” (Way, 1991, p.124)

Max Black originated the interaction view (Black, 1979) and Way summarises the main points of the interaction view of metaphor as follows:

- metaphor involves entire systems of assumptions and ‘commonplaces’ which are associated with the terms involved;
- that the metaphorical process works like a filter, with the associated ideas of the secondary subject (vehicle) hiding, highlighting and organizing aspects of the primary subject ;
- understanding metaphor often involves a shift in meaning ;
- metaphor cannot be reduced to any literal statements of comparison, and ;
- metaphor can actually create similarity between previously dissimilar ideas.

(Way, 1991, p.48)

Way acknowledges that there are criticisms of the interaction view, however, she believes that despite these “the interaction view has been the most widely accepted and influential view of metaphor.” (Way, 1991, p.50) Consequently, Way’s original theory, the DTH theory, is developed as a variation and improvement on the interaction view theory. The DTH theory is thus purported to explain the results of empirical tests where all other theories fail.

Way's approach to metaphor and knowledge representation "is within the framework of, and as an extension to, an existing theory for processing natural language, namely, that of *conceptual graphs*." (Way, 1991, p.96) Way sees Sowa’s conceptual graph (CG) theory of knowledge representation as an advanced system, which can be adopted to represent metaphorical features of language. The primary difference between DTH and CG hierarchies is that DTH’s are, as the name suggests, *dynamic*, while CG hierarchies are represented as static structures. “... the boundary of what is literal and figurative is constantly shifting: we all know that today’s metaphor may be literal tomorrow and vice versa. ... Thus, if we are going to use the notion of a type hierarchy to explain literal and figurative uses of language, it will have to be a *dynamic* type hierarchy.” (Way, 1991, p.23) This results in “a hierarchy which is generative and constantly changing over time.” (Way, 1991, p.111)

The DTH theory uses standard concepts from graph theory: “A type hierarchy is a network of concepts which are organized according to levels of generality. ... So the links connecting the supertypes and subtypes of the semantic network represent going from an instance of a supertype to a more specific instance of that supertype. ... an instance of a subtype entails that it is also an instance of the corresponding supertype.” (Way, 1991, p.21) The DTH is a generalisation hierarchy where the entities are categorised based on the common attributes. In a generalisation hierarchy, the higher-level class (supertype) shares the common attributes of the lower level class (subtype), while subtypes inherit all the properties of its supertype.

“The DTH theory holds that the similarity found between the tenor and vehicle of a metaphor⁹ is not an intersection of their properties; rather, it is generated by finding a common and more abstract *supertype* that the two share.” (Way, 1991, p.40) The supertype is a generalisation of the attributes of the connected subtypes. “In metaphor, what is common between the vehicle, and tenor is not an intersection of a list of features at the level of the tenor and vehicle, but a supertype, which is higher up on the semantic hierarchy and under which aspects of both the vehicle and tenor domains fall. Furthermore, which supertypes are chosen, assuming that there are several in common, is a function of the context and the direction of the attribution of the metaphor, that is, the metaphor is attributing features from the vehicle domain by abstracting them to a common supertype and then using that supertype to pick out the corresponding features of the tenor.” (Way, 1991, p.129)

Under the DTH theory, metaphor operates through a higher-level supertype that connects the tenor and vehicle types. If this supertype is not found to already exist in the type hierarchy, it is dynamically created, just as in some cases "metaphor creates the similarity" rather than describing some existing similarity (Black, 1979). Two separate clusters of schemata thus become associated through a supertype and this allows a migration of concepts from the vehicle to the tenor. This exemplifies what Black terms “systems of associated commonplaces” (Black, 1979): the complex and far reaching web of associations and meaning that imbue every metaphor.

⁹ Here, the terms ‘tenor’ and ‘vehicle’ can be reasonably substituted for ‘target’ and ‘source’ respectively.

Way provides as an example the metaphor ‘the car is thirsty’. This metaphor “involves a violation of a constraint, in that thirsty is an attribute of an animal, not a vehicle. The new supertype, *Mobile-entities that require liquid*, can have both tenor and vehicle fall under it without violating any semantic constraints.” (1991, p.130)

"Note that the set of properties of the supertype will be smaller than those of its subtypes; in this way the supertypes are able to act as filters on lower level concepts. ... The supertypes VEHICLE picks out one aspect of a car, while another supertype, say STATUS-SYMBOL, would pick out or filter different aspects." (Way, 1991, p.161)

A drawback of the DTH theory is the reliance on Conceptual Graphs (CG's). CG's allow a precise level of description and are well accepted in the academic community. However, they are difficult for the layperson (even a well-educated one) to produce and understand without a considerable amount of training (refer to section 2.5.5). Furthermore, the inclusion of CG theory does not appear to be *necessary* to the development or description of DTH theory and the intercoupling between these theories appears somewhat contrived.¹⁰

Another potential weakness of the DTH theory is the fact that, while attempting to objectify our understanding of metaphor, the development of DTH's is a rather heuristic process, requiring “background knowledge” and “reasoning systems” in order to create an appropriate hierarchy (Way, 1991, p.143). However, given the nature of linguistics, it may be unrealistic to think that our understanding of metaphor could be systematised much further. Consequently, the DTH theory offers a very useful, systematic, formal and codifiable understanding of metaphor.

3.3.1 Support for Way's Theory

Despite Way's carefully thought out theory of metaphor, and its publication as a book in 1991, there is scant reference to the DTH theory of metaphor in the literature. This could give rise to the suggestion that the theory has, by default of not being accepted and widely referred to by the cognitive science research community, been *discounted* by

¹⁰ As the focus of this thesis is on the intrinsic hierarchical nature of metaphor, the decision by Way to represent the DTH theory using Conceptual Graphs has no bearing on the outcomes of this thesis.

this community. In this case, its use as the basis for the development of new academic work may be, at best, ‘courageous’.

However, this theory appears to have new support through the recent publication of a carefully thought out paper by Cornelissen (2005). Cornelissen appears to, unknowingly, have reinvented Way’s theory (or one very close to it). Since Cornelissen’s work so closely relates to Way’s work, the lack of any reference to Way in such a well-referenced work can only mean that Cornelissen was not aware of Way’s contribution.

Cornelissen states “... current perspectives are flawed and misguided in assuming that metaphor can be explained with the so-called comparison model.” Like Way, Cornelissen cites the overwhelming research evidence that refutes this model.

In response, Cornelissen outlines an alternative model for understanding how metaphor works, which he calls the “domains interaction model”. This model “... suggests that metaphor involves the conjunction of whole semantic domains in which a correspondence between terms or concepts is *constructed* rather than deciphered and where the resulting image and meaning is *creative*.” (Cornelissen’s italics). The important features of the metaphor, Cornelissen claims, are *emergent* and are conceptually connected to the source and target conceptual domains.

This is remarkably similar to Way’s suggestion that metaphor operates through a dynamically created, higher-level supertype that connects the lower level, source and target concepts. Yet, Way’s work appears to be the more formal of the two approaches since it ties a theory of metaphor to an understanding of concept types and hierarchies, providing a more objective analysis and explanation of the mechanisms that underlie the operation of metaphor.

The fact that Cornelissen has independently arrived at, what can be argued to be essentially the *same* conclusions as Way, provides support for Way’s theory and legitimises its use as a basis for further research.

3.4 Summary

“When (men) use words metaphorically; that is, in other sense than that they are ordained for, (they) thereby deceive others ... Such (inconsistent) names can never be true grounds of any ratiocination.”

Hobbes, from Leviathan

This chapter has presented a critical review of the research in the application of metaphor to IT systems. Commonly used EA metaphors have been identified. It has also been revealed that metaphors form a hierarchy extending from the most highly concrete to the most conceptual metaphor. More formally, metaphor forms part of a dynamic type hierarchy.

The deliberate application of metaphor to computer systems modelling is popular and widely encouraged. Yet, the case for the value of metaphor in the realm of IT systems modelling is far from proven. While there is surprisingly little empirical research in this area, the available results showing the value of metaphor in information systems design are far from convincing. On the other hand, the *pitfalls* of using metaphor are easily demonstrable, and it *has* been observed that the use of metaphor for representing IT entities is highly problematic. In fact, there are distinct disadvantages to the use of metaphor, especially when the information scope is wide. “The view that a system must stick to a metaphorical representation as closely as possible is one we believe to be mistaken: the system should improve upon the metaphor, not be bounded by it.” (Hammond and Allison, 1987, p.88)

The continued use of metaphor in computer discourse is inevitable and its elimination is (literally) unthinkable. The question remains as to whether there is a better way of developing and using metaphor for systems modelling that obviates the shortcomings of contemporary methods. Before such a method can be developed, it is necessary to examine the nature and requirements of system modelling. This is the topic of the following section.

4 THEORETICAL PRINCIPLES FOR THE DEVELOPMENT OF UNIFIED EA MODELLING LANGUAGES

4.1 Introduction

The development of an effective *unified* EA modelling language would overcome significant shortcomings of contemporary EA approaches and would thus improve the business value of any EA. Yet, as we have seen, the development of such a language, using traditional and direct approaches to the problem, has remained elusive.

There are two conditions for the development of an effective unified model. Firstly, the unified model must be capable of representing the semantics of the various sub-domains that are being modelled (albeit, at a higher level of abstraction). Secondly, the unified model must present a strong ontology so that the model can be interpreted unambiguously. Any well-defined and formalised modelling language can provide a strong ontology. However, no current modelling languages can cover the *semantic* breadth needed to describe the different types of enterprise systems, without inordinate complexity.

In this chapter, the problem of developing an enterprise model that is semantically unified is attacked. The structure of this approach is as follows. Firstly, a case will be presented for the argument that metaphors are models. It is then shown that these metaphors are part of a concept type hierarchy, in keeping with the dynamic type hierarchy (DTH) theory. This is illustrated as a Venn diagram in Figure 13. We previously saw that the DTH theory can be used as a way to explain the *mechanisms* by which metaphor operate. In this research, the theory is extended in order to *generate* metaphors that can be used as EA models.

In the next chapter (Chapter 5), this model is used to develop an ontology that can be applied to formally describe any multiple EA domains. This ontology is then formalised and codified to produce a unified EA modelling language.

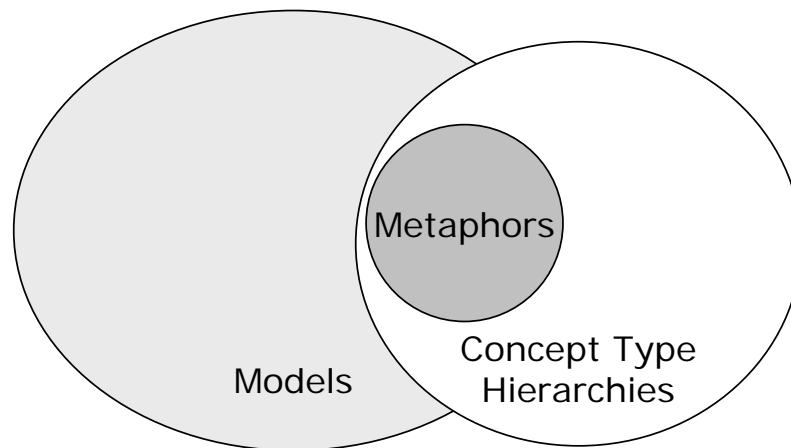


Figure 13 - The Relationship between Models, Metaphors and Concept Type Hierarchies

4.2 Models and Metaphors

As our understanding of metaphor has developed over the years, the links between models and metaphor have become more apparent. Certainly, this is quite clear in the case of linguistic models. Black was one of the first to make the case for the connection between models and metaphor in language and philosophy (1979), and in later years, Black becomes further impressed "by the tight connections between the notions of models and metaphors. ... Every metaphor is the tip of a submerged model." (1979, p.31) More recently, these early notions on the connection between models and metaphors have been extended. Lakoff, for example, demonstrates that mathematical models are also metaphors (Lakoff and Nunez, 2000). Moreover, according to Roussev and Rousseva, the mechanics of modelling are the same as metaphor and modelling evokes the same cognitive processes as metaphor. In fact, "Models are always grounded in a kernel metaphor, or metaphors, and hence stimulate many of the same cognitive processes." (Roussev and Rousseva, 2004)

Interestingly, the definition of a model - "A model is a representation of something else ..." ISO/ANSI, as quoted in (Szegheo, 2000), is remarkably similar to the definition of a metaphor as an "image that represents one thing as something else in order to explain it better ..." (Johnson, 1994).

In fact, an empirical argument can be made for the assertion that metaphors *are* models. According to Herbert Stachowiak (as referenced in (Ludewig, 2003)) an

artefact must satisfy the following three criteria in order for it to be identified as a model:

- Mapping criterion: there is an original object or phenomenon that is mapped to the model.
- Reduction criterion: not all the properties of the original are mapped on to the model, but the model is somehow reduced. On the other hand, the model must mirror at least some properties of the original.¹¹
- Pragmatic criterion: the model can replace the original for some purpose, i.e. the model is useful.

It will be recalled that a metaphor connects a source domain to a target domain. This satisfies the first criterion. Also, when metaphor is used, only certain characteristics of the source are mapped to the target, and this depends upon the context in which the metaphor is being used (Way, 1991). Thus, the second criterion is satisfied. Finally, we use metaphor in order to explain something better (Johnson, 1994), especially when that something is new and novel (Ludewig, 2003). Therefore, metaphor satisfies all three of Stachowiak's criteria defining a model. We can conclude that metaphor is a type of model. More specifically, metaphor is a type of conceptual model (Allen, 1997).

It can be observed that the labels commonly given to enterprise systems are actually metaphors (and by induction, models). Take as an example, an enterprise system labelled 'Customer Relationship Management System'. If this were simply a label, and not a metaphor, then there would be no significant effect in changing this label to 'Customer Exploitation Management System', as long as there were no changes made to the actual systems to which this model refers. However, in fact, this label change fundamentally changes the perceptions that will be generated around this system. A completely different set of "associated commonplaces" (Black, 1979) has been set up, because these labels are not simply literal: they are metaphors which give rise to supertypes which encapsulate much of the understanding of what the system is and

¹¹ It must be noted, that this is contrary to some popular notions of models. For instance, Ghyczy suggests that models "exhibit a one-to-one correspondance" with their source and that you can "transfer everything you know about the source domain into the target domain" if you have a good model GHYCZY, T. V. (2003) The Fruitful Flaws of Strategy Metaphors. *Harvard Business Review*, 86-94.

how it operates. Consequently, an “entire web of associations and implications” (Way, 1991, p.36) has been altered.

In fact, the metaphorical approach to organisational modelling is founded upon the observation that the reality of organisational structures are not concrete, but abstract, indefinite, and perhaps even undefinable, entities. There is no objective reality of an organisation. The significant structures within an organisation are actually social constructs, and as such, they are imbued with cultural and social meaning, presenting differently to every person according to their immediate concerns. In modelling organisations, we are not simply developing abstractions of an objective and concrete structure. Rather, we are saying ‘this is how we wish the organisation to be understood’. For instance, ‘we want our enterprise to be seen as developing relationships with customers, not exploiting them’. As Black says, the metaphor actually helps constitute aspects of reality (Black, 1979). We are creating meaning through the development of metaphors for these structures that have a generative quality.

4.3 Model Hierarchies

As shown in Section 3.3, metaphor can be considered part of a dynamic type hierarchy. For the purposes of the argument presented here, the types in this hierarchy will be restricted to the domain of enterprise models. Note that, since it has been shown that metaphors are a special type of model (in Section 4.2), this restricted domain may still include metaphor.

Within any enterprise, there is a set of models representing various levels of abstraction. At the highest level, is the model of the enterprise itself: “EAs are systems of systems, there is an emphasis on a higher level of conceptual modeling.” (Kaisler et al., 2005). This enterprise model may, or may not be formalised. At lower levels of abstraction are more detailed models such as system and function models. Recall from section 3.2.6 that larger scale structures tend to be modelled using more general, 'conceptual' metaphors while smaller scale structures tend to be modelled using more specific, 'concrete' metaphors. This is because enterprise system models can be formed into a generalisation hierarchy, where the higher-level classes (supertypes) share the

common attributes of the lower level classes (subtypes), while subtypes inherit all the properties of its supertypes.

Figure 14 shows an enterprise system hierarchy where Enterprise A is the universal supertype. According to Way, "metaphor cannot take place by comparing the properties of independent systems on the same level", i.e. at the same level of abstraction. "In order to properly make the comparison, we must search for higher order concepts. Comparison of properties on one level only make sense as a comparison with respect to a common, more abstract, higher level property." (1991, p.144)

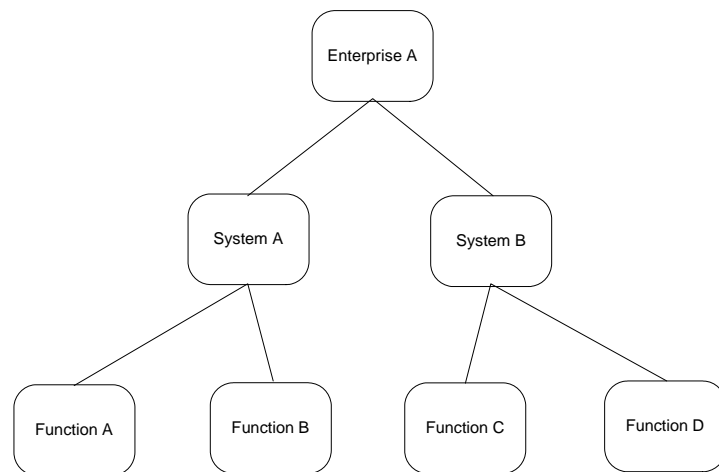


Figure 14 - Enterprise System Hierarchy with Enterprise as Global Supertype

Therefore, it would be possible to create a metaphor at the same level of abstraction as Enterprise A that could be used to describe features of each of its component structures (the System A, System B, Function A, Function B etc). However, as every enterprise is unique, this metaphor may not be relevant to other enterprises or their component structures.

On the other hand, an enterprise metaphor at a *higher* level of abstraction than the type 'enterprise' would be relevant to all enterprises and their component structures. In fact, it would serve as a unified concept that could be applied to any of its sub-models. This

is illustrated in Figure 15. The task remains, to identify, or invent, a supertype of all enterprise systems models.

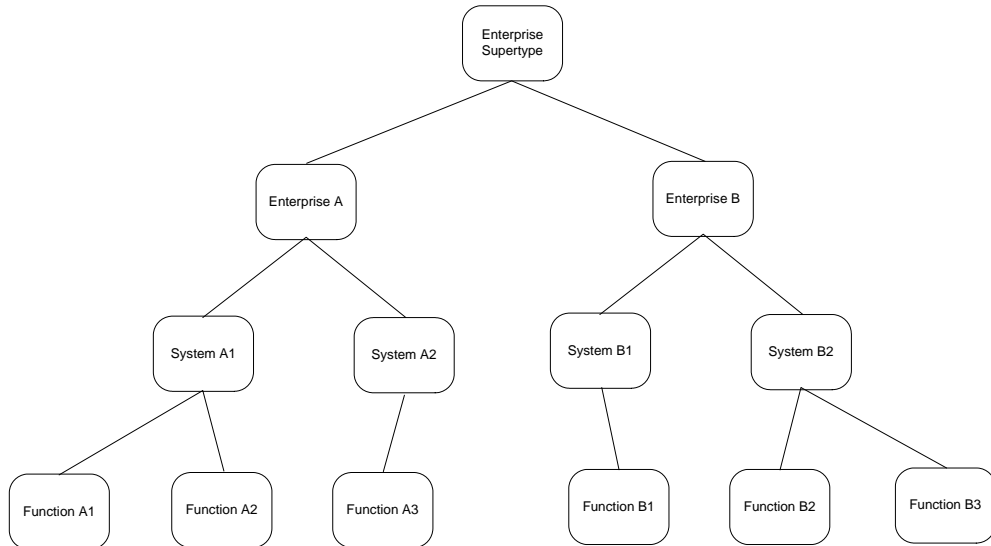


Figure 15 - Enterprise System Hierarchy with New Enterprise Supertype

Identifying the level of abstraction of a concept is not an easy undertaking (Biemans et al., 2001). However, if we think about concept hierarchies in terms of structures it becomes easier. For instance, an interface can include a button, but a button cannot include an interface. Therefore, the concept of an interface is at a higher level of abstraction than the concept of a button. Similarly, a system can include a program, but not vice versa. Therefore, the concept of a system is at a higher level of abstraction than a program.

There are any number of abstract concepts that could be used as a metaphor for enterprise systems. These include existing metaphors that have been used to describe organisations including the description of organisations as machines, organisms (Fayad et al., 2002), brains, nervous systems (Fayad et al., 2002), cultures and political systems (Morgan, 1996), learning organisations (Senge, 1990), organisational identity (Cornelissen, 2005), organisational mind (Sandelands and Stablein, 1987) and organisational memory (Walsh, 1995).

Some of these metaphors use source objects that are at a lower level of abstraction than the enterprise structure. Therefore, they are not suitable as a unifying metaphor

for enterprise systems. For example, the description of an organisation as a machine has almost disappeared from view, and yet has had a profound influence that has left us with residual concepts such as the organisational “structure” concept (Morgan, 1996). While this metaphor may still describe certain aspects of the organisation, it is a concrete metaphor and, as a result, is too confining to describe all aspects of all systems. For instance, it cannot describe the ability of an organisation to learn, regenerate or develop purpose. The metaphor of the ‘learning organisation’ is however more conceptual and flexible, explaining perhaps its greater popularity as an enterprise metaphor.

In Chapter 5, the metaphor ‘an enterprise is a society’ is identified as an appropriate unifying metaphor and is adopted and developed as the basis for a unified systems language.

4.4 A Methodology for Developing Unified EA Modelling

Languages

The approach presented in the previous chapters shows how an understanding of type hierarchies can be used to develop a unified enterprise modelling language. In some real-world enterprise situations, there will be no existing, formal models at the EA level of abstraction. In this case, the new unified models will be developed from scratch based on available information from a variety of possible sources. In other enterprise situations, EA models will already exist, but they will not be unified models. Most likely, they will have been developed using a variety of methodologies and languages, some formal and some informal. In this case, the models can be redeveloped as unified models through an interpretive approach where the syntax and semantics of the current models are translated into the new constructs.

In the aforementioned situations, this methodology satisfies the need for unified models that extend across all enterprise domains at the highest levels of abstraction. However, it was previously noted (in Section 1) that one shortcoming of contemporary system architecture approaches is that the languages that are used to describe EA’s are restricted to a narrow range of abstraction, and that there are benefits to having a language that can describe multiple levels of abstraction ranging from strategy to implementation. It was also noted that there is a trade-off between the potential of a

language to be used to model a wide scope of functionality, versus its use to model a wide range of abstraction levels. Furthermore, it was stated that, if priority is given to providing a wide modelling scope, then some sort of mapping must be provided between the different levels of abstraction.

This challenge is addressed in the following way. Firstly, the unified language is designed for extremely wide scope so that it can be used to model all enterprise systems at the highest levels of abstraction. This quality is assured by developing an enterprise metaphor that is based on the enterprise system supertype. The description of lower levels of abstraction is then described traditionally, using existing, domain specialised, modelling languages. However, the unified language is used to *structure* the models developed at each level of refinement (the opposite of abstraction), thus ensuring consistency between the different abstraction levels. This can be achieved because, in the same way that the enterprise model supertype is used as a metaphor at the enterprise level of abstraction, it can also be used as a metaphor for any of its component structures at lower levels of abstraction. For example, if the selected enterprise model supertype is 'game', then all of the systems within that enterprise are viewed as game structures: the enterprise is a game, the system is a game, the interface is a game, and the interface elements (if we assume this is the lowest level of abstraction) are all elements of the game.

Note that the language developed using this approach is not intended to *replace* other domain specific languages and methods. At levels of description below the enterprise level, where more detail is needed to describe the system's characteristics, this unified language would lack the semantic strength required to describe these systems fully. Rather, the unified language is designed to augment the existing domain specific languages and methods, in order to ensure that the enterprise systems' structural integrity is maintained throughout the development of more detailed levels of abstraction by aligning these structures to one, over-arching metaphor. This concept is illustrated in Figure 16 where the unified language is referred to as LEAN (Lightweight Enterprise Architecture Notation), the unified language that is developed in later sections of this thesis.

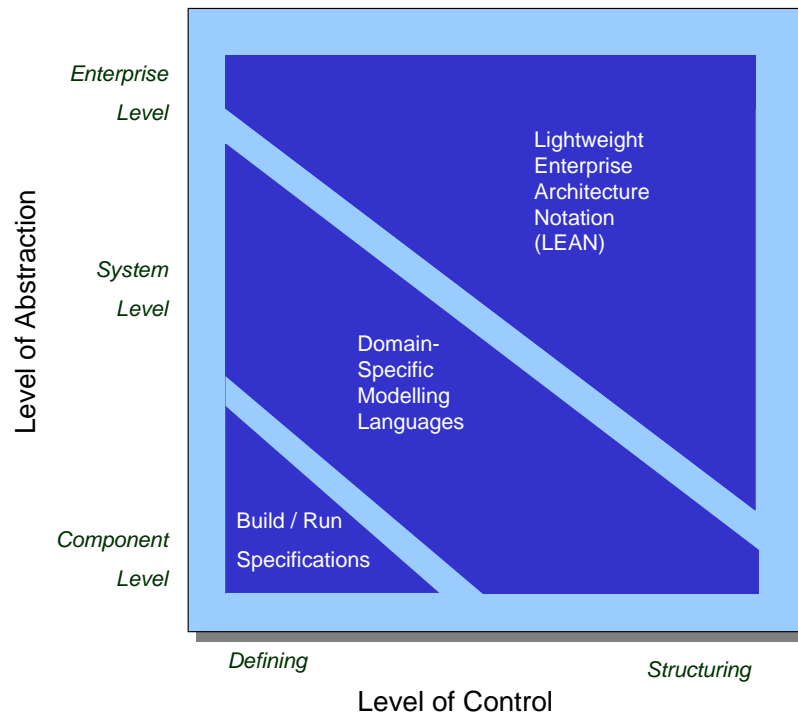


Figure 16 - The Applicability of LEAN at Various Levels of Abstraction

Figure 17 shows the stages within the methodology that is used to develop and apply a unified modelling language. The first stage of this methodology, ‘Identify an enterprise metaphor’, is covered in the following Section 4.5. In this section, a societal metaphor is selected as an example of an enterprise supertype that appears well suited to the task of EA modelling.

The second and third stages, ‘Specify and formalise the language’ and ‘Codify the enterprise metaphor’ will be covered in Chapter 5. In this chapter, the societal metaphor will be developed into an ontology, and that ontology will then be formalised and codified to produce a high-level, unified, EA modelling language.

4.5 An Enterprise Metaphor

Based on an understanding of both the role, and nature of metaphor, the proposed methodology provides a means for developing high-level metaphors that can be used to describe multiple EA domains and subsystems.

The societal metaphor was previously provided as one example of a metaphor that can be used to describe enterprises. This metaphor will now be further developed and

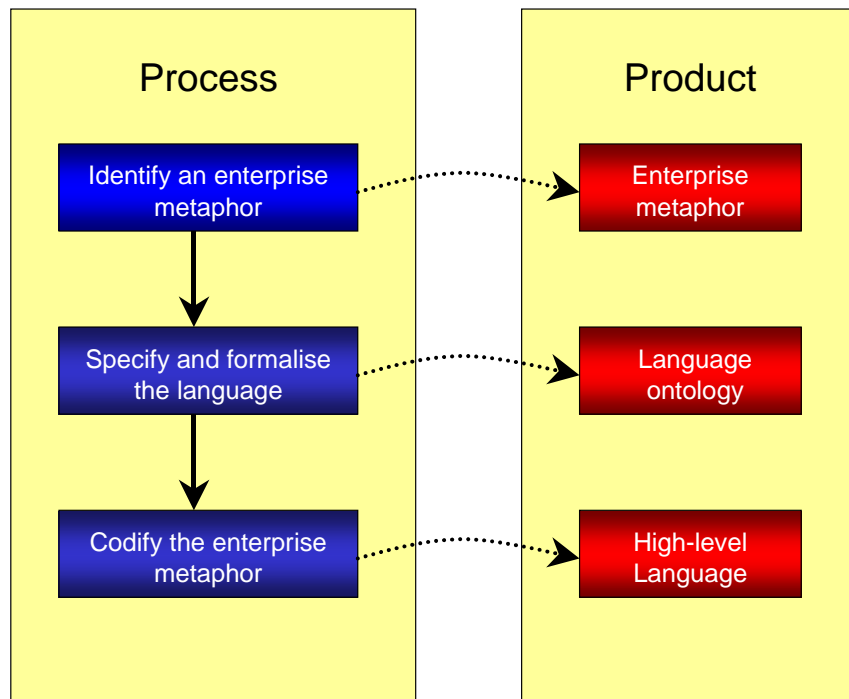


Figure 17 - Methodology for Developing and Applying a Unified Language

analysed, and then, in the following chapter, developed into a formal EA modelling language.

Perhaps one of the conceptual structures most familiar to humans is that of human society. Society is a larger scale structure than an enterprise, and accordingly, it can be used as a supertype of all other types in the enterprise system type hierarchy. That is, a society-sourced metaphor will be at a higher level of abstraction than any of the models that describe an enterprise or its components. Therefore, the concept of society has the semantic breadth to describe all enterprise systems.

Is a society-sourced metaphor a good choice for modelling enterprise systems? It is clear that all enterprises are social phenomena and that computer systems are socially embedded phenomena. In lieu of a social context, computers can have no meaning or value, and outside of language, computers do not even exist (Winograd and Flores, 1987, p.78). It is also noted that many systems failures occur, not because of technical issues, but because the relevant socio-political concerns were not evaluated and addressed (Highsmith, 2002). A society-sourced metaphor is likely to be stronger than many other metaphors for modelling these aspects of the enterprise.

It should also be noted that the societal metaphor has *already* been widely adopted as a metaphor to describe complex, autonomous and interacting computer systems, i.e. intelligent multiagent systems (Kolp et al., 2005) (Wooldridge, 2002). In adopting this metaphor, Wooldridge (2002) describes the trend "... away from machine-oriented views of programming toward concepts and metaphors that more closely reflect the way in which we ourselves understand the world." Ho et al (1986) have used the society metaphor to model office information systems. Bernard (2004 p. 48) refers to enterprises as "social enterprises", while De Geus (1997) believes that the prevailing thinking and language of economics is contributing to the failure of corporations, because they "forget that their organizations' true nature is that of a community of humans." And Peter Drucker (1992), the highly respected 'guru' of modern management, refers to "The society of organizations".

The choice of a societal metaphor may also prove to be more *persuasive* than other metaphors since, by presenting an overt societal metaphor, the computer is "positioned more as a social actor than as a machine or 'neutral tool'." (Marakas et al., 2000) This approach is perhaps justified by the observation that, "Information technology is arguably, like society itself, an abstract concept." (Marakas et al., 2000)

Finally, the societal metaphor provides a concept that meets the criteria set out by Proper et al, for the identification of concepts that are effective in describing a given modelling domain: it must be simultaneously simple and rich, it must be capable of describing anything that needs to be modelled within that domain (in our case, EA), and it must be understandable by any interested party (Proper et al., 2005).

Thus the 'pragmatics' of this metaphor (as defined by Biemans et al¹²) appear to be strong, especially as any society based concepts and nomenclature are likely to be very familiar to all users of the enterprise architecture. While there *are* other metaphors that would satisfy the semantic requirements of a unifying supertype, the concept of society is perhaps a more promising metaphor for this problem, since it has a particular relevance and efficacy for enterprise architecture modelling.

¹² Pragmatics: A measure for the degree to which models expressed using this metaphor will succeed in being interpreted by the audience as intended by the creator. BIEMANS, F. P. M., LANKHORST, M. M., TEEUW, W. B. & WETERING, R. G. V. D. (2001) Dealing with the Complexity of Business Systems Architecting. *Systems Engineering*, 4, 118-133.

What does a model of society look like? One such model has been developed by Giddens (1984). According to Giddens' Theory of Structuration, there is interdependency between humans (Actors) and societal structures (Resources and Rules) that is manifest through specific Actions. Thus, Giddens provides a lexicon of Actors, Resources, Rules and Actions that can be used to describe societies.

The notion of an Actor is extended here to include, not only individuals, but also any Agent that can exert power in order to produce an effect. To this end, the terms Actor and Agent are used interchangeably. Resources are “structured properties of social systems, drawn upon and reproduced by knowledgeable Agents in the course of interaction.” Resources are of two types. Allocative resources are material resources involved in the generation of power and derive from human dominion over nature. Authoritative resources are non-material and derive from the capability of harnessing the activities of human beings (Walsham and Han, 1991, p.84). Rules refer to the sanctioned modes of conduct. Finally, an Action is an activity that is performed. "Structures, as 'rules and resources', do not do anything, but they have their effect through being known and used by actors." (Parker, 2000)

This provides us with four concepts (Agents, Resources, Rules and Actions) that can serve as the foundation of a unified EA ontology. For convenience, we refer to this modelling language as the Lightweight Enterprise Architecture Notation (LEAN).

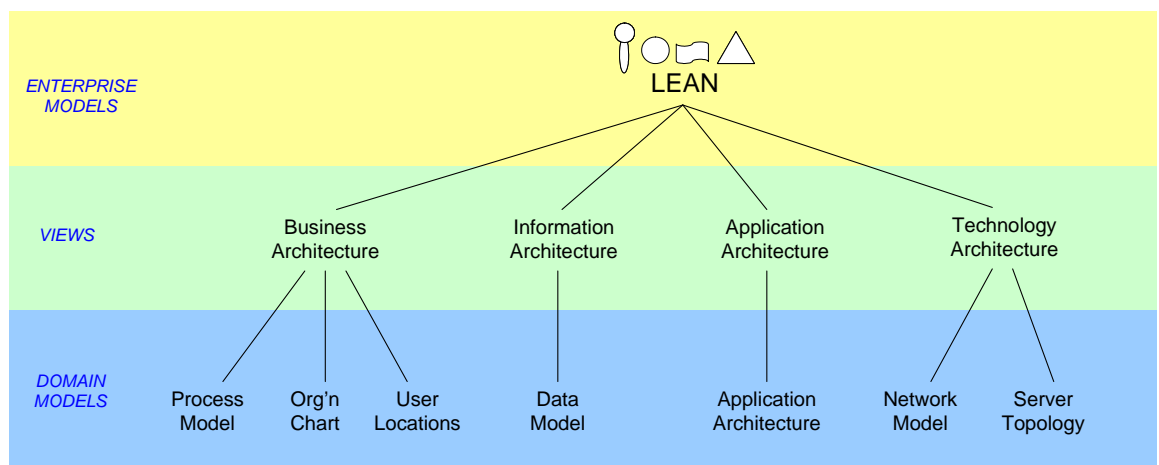


Figure 18 - LEAN in relation to EA views and domain specific models

Figure 18 illustrates how this language would be situated in terms of its level of abstraction and relation to the commonly accepted EA domains (or views) and domain specific modelling notations.

4.6 Summary

In the previous chapters, a review of linguistic, cognitive and information systems theory was presented in order to provide a framework within which current theories and approaches to EA modelling can be situated. Within this chapter, this understanding was used as a foundation upon which to develop a novel method for developing a unified EA modelling language. The theoretical principles for this approach were expounded and a methodology for developing a unified EA language was described.

In the following chapter, this methodology is applied and a high-level unified EA modelling language is generated.

5 THE LEAN ONTOLOGY

Newly designed modelling languages should be based on reliable ontology's (metamodels with semantic rules), which will ensure the required consistency and determine the expressive power of the modelling as required by the desired enterprise view. (Noran, 2003)

5.1 Introduction

The theoretical principles used for developing unified EA modelling languages that have been developed in the previous chapter will now be applied to develop on example of a unified EA modelling language. To do this, a candidate metaphor is developed as an ontology, which is then codified as a graphical modelling language.

An ontology is a specification for the representation of some abstract, simplified view of the world (Gruber, 1993). It is a shared conceptualisation that provides a common understanding of some domain. Ontologies consist of a set of categories or ideas in the world (concepts) along with certain relationships between them (Hirst, 2003).

In this case, we are viewing the world from the perspective of an enterprise architect and the relationships that we are interested in are those that support our understanding of enterprise systems.

Ontologies are important because “They provide a shared and common understanding of a domain that can be communicated between people, and heterogeneous and widely spread application systems.” (Pinto and Martins, 2004) “An ontology includes a catalog of terms used in a domain, the rules governing how those terms can be combined to make valid statements about situations in that domain, and the *sanctioned inferences* that can be made when such statements are used in that domain.” (Knowledge Based Systems, 2004a)

In this chapter we design and develop an ontology based on the metaphor, ‘an enterprise is a society’. This ontology will be referred to as the LEAN (Lightweight Enterprise Architecture Notation) ontology. The goal of this design work is to define the vocabulary and semantics that will be used in describing enterprises using the concept ‘enterprises that exemplify society’. This will then serve as the basis for developing a graphical language for describing EA’s. The development of this ontology

will therefore allow the intended meaning of constructs that are developed using this language to be clearly communicated between different agents.

Ontologies can be developed at a number of levels ranging from highly formal to informal (Fox and Gruninger, 1998). The LEAN ontology is developed as an informal ontology. That is, the definitions are expressed using natural language. It is not designed to support *automatic* integration with other ontologies or to be computationally executable. Further formalisation of the LEAN ontology *could* allow this in the future.

A number of different versions and iterations of the LEAN ontology have been developed as part of this research. These versions were initially tested by transforming existing, public domain, EA models into LEAN models. The effectiveness of these LEAN models was then informally evaluated. These evaluations provided important feedback on the ease-of-use, semantic richness and clarity of each iteration of the ontology. Based on this feedback, these early versions of the ontology were then progressively developed and refined until the optimal LEAN ontology described in this chapter was settled upon. They included the following variations:

- Ontologies where the relationships were non-directed
- Ontologies where various of the societal concepts were developed to be represented as relationships rather than nodes (for example, a version was developed where the Rule concept was used to describe the relationship between the other concepts)
- Ontologies where the relationships were restricted to particular forms. For example, any two societal concepts could only be related via the Action concept.
- Ontologies where the meaning of the relationship depended upon the nodes that were being connected by it.

While all of these approaches were successful to some extent, the LEAN ontology described below was found to provide the most flexibility, semantic richness and clarity, and the greatest ease-of-use of all of the variations tested.

5.2 The Ontology Development Methodology

The methodology used in developing this ontology is loosely based on the approach developed by Uschold and King (1997), and also discussed by Pinto and Martins (2004), who refer to it as the “Enterprise” methodology. The components of the Enterprise methodology are as follows:

1. Identify the purpose and scope of the ontology:
 - a. Why is the ontology being built?
 - b. Who will use the ontology?
 - c. How will it be used?
2. Construct the ontology:
 - a. Ontology capture.
 - b. Ontology coding (formalisation and implementation).
 - c. Ontology integration (re-using appropriate knowledge from existing ontologies).
3. Evaluate the ontology.
4. Document the ontology to support knowledge sharing.

While the Enterprise methodology provides a useful starting point, the ontology development methodology used in this research departs from this methodology in a number of key respects, which will now be described.

Firstly, identification of the purpose and scope of the ontology is not performed *explicitly* as part of the LEAN ontology development. In the case of the *Enterprise* methodology, this stage is used to expound upon the problem to be solved: i.e. why do we need a new ontology, and what are the boundaries of the domain we wish to describe? In case of *this* research however, the purpose and scope of the desired ontology has *already* been clearly defined and elaborated upon in previous sections of this thesis. Instead of repeating this work, a summary of these objectives within the current context is provided in order to provide a clear focus for the development of the remainder of this chapter.

Another area in which the ontology development methodology used in this research departs from the Enterprise methodology is in relation to step 2: Construct the ontology. While the steps of ontology capture, coding and integration may be viewed as *conceptually* discrete, the artefacts that are generated by these steps are, in this case, interwoven and interdependent. Thus, in describing, for example, a certain concept, its definition, representation and references to integrated knowledge may be grouped together for clarity, or presented in various sequences.

Formal integration with other ontologies is outside the scope of this research. Compared to LEAN, other well known enterprise ontologies (such as TOVE (Fox et al., 1993), the Enterprise Project (Uschold et al., 1997), and the IDEF Ontologies (Knowledge Based Systems, 2004a)) contain a far larger number of, more granular, concepts. The LEAN ontology, on the other hand, has been deliberately kept at a conceptually high-level in order to aid understandability and efficacy to the enterprise-modelling problem. However, developing the mechanisms and procedures for integrating these ontologies with LEAN is suggested as an area for further research (refer section 8.2).

Furthermore, it should also be noted that, in line with the methodology described in Section 4.4 and illustrated in Figure 16, the process “Identify an enterprise metaphor” *precedes* development of the language ontology. In the ontology capture stage of the *Enterprise* methodology the key concepts and relationships in the domain are identified and defined. In our case, the key concepts and relationships that will be used in the domain are derived from an understanding of the selected metaphor source, which is, in this case, ‘society’. The metaphor is considered appropriate if these concepts and relationships map well to the metaphor target (Enterprise Architecture), and this is supported by the application of the Dynamic Type Hierarchy Theory for the identification, or invention, of potential metaphor candidates. Thus, the key concepts and relationships in the domain will be constrained by the metaphor that has been selected.

Evaluation of the LEAN ontology is carried out through the three research studies that are documented in Chapters 7.1, 7.2 and 15. In these studies, important criteria for assessing the value of the LEAN ontology are assessed.

Lastly, *documentation* of the LEAN ontology is the focus of this chapter. Appropriate documentation of the ontology supports effective knowledge sharing (Uschold et al., 1997) and so the documentation provided here is designed to provide the essential information required to support knowledge sharing without becoming superfluous.

5.3 The LEAN Ontology

5.3.1 Purpose and Scope of the Ontology

Earlier in this thesis, it was demonstrated that a language is needed that allows the generation of EA models spanning multiple IT domains. In fact, the weakness of contemporary approaches to EA modelling is that numerous languages are required in order to model EA's formally and this puts a high cognitive load on both modellers and users. Thus, the use of a human-centric, unified language will make the development of EA's more efficient, and will make those EA's more effective since both technical and non-technical stakeholders alike, will be better able to understand such models. Indeed, the audience for such models extends well outside of the domain of technical specialists who are directly involved in architectural activities, and often includes:

- “C-level” executives (e.g. the CEO, CIO and CFO)
- Senior executives and managers who represent business interests
- IT personnel at all levels
- Shareholders and investors

Clearly, it cannot be assumed that these important EA stakeholders have well developed architectural skills. Yet, they are often important contributors to the development of an EA, and it is often vital that the outputs of an EA program can be communicated to these parties.

Figure 16 shows the process by which this language will be developed: a crucial step in this process is the development of a formal ontology.

5.3.2 Ontology Construction

It was noted, in Section 4.5, that the metaphor ‘an enterprise is a society’ is a promising metaphor for describing enterprise systems. It was also noted that Giddens has

developed a model of society that can be used as a basis for formalising the source of this metaphor (society). Giddens model is based on four primary concepts: Agents, Resources, Rules and Actions. This process is analogous to the ‘ontology capture’ process described in the Enterprise methodology. These concepts thus become the key concepts for our ontology.

5.3.2.1 Formalising the Enterprise Metaphor Concepts

The four key concepts of Agents, Resources, Rules and Actions were briefly defined in Section 4.5. These concepts are defined more rigorously in Table 6.

Agent	
Definition	An Agent is an entity that can exert power in order to produce an effect. In relation to IT systems, the immediate effect is the exchange of information.
Description	In the LEAN ontology, the effects produced by Agents are referred to as ‘Actions’.
Examples	Agents may be: <ul style="list-style-type: none"> ▪ People ▪ Roles ▪ Organisations ▪ Communities ▪ Nation-states ▪ Systems
<i>Notes</i>	In the case of temporal events, the Agent may be the system itself. In all other cases, the Agent is the entity that triggers a system event. Any effectual Agent will have an ‘area of concern’ within the system. This area of concern can be used as the basis for developing an Agent related view of the system.
Resource	
Definition	A Resource is a structured property of the modelled system that can be consumed or produced by one or more Agents.

Description	A Resource represents a natural constraint within the system.
Examples	Resources may be: <ul style="list-style-type: none"> ▪ Raw materials ▪ Systems ▪ Documents ▪ Images ▪ Services ▪ Agents
<i>Notes</i>	When Agents are used to signify constraints on the system, they can be represented as Resources.
Rule	
Definition	A Rule defines a sanctioned mode of conduct.
Description	A Rule regulates the type of Actions that may take place within a system.
Examples	Rules may be: <ul style="list-style-type: none"> ▪ Physical constraints ▪ Logical constraints ▪ Legal and regulatory compliance ▪ Standards and guidelines ▪ Business goals or objectives
<i>Notes</i>	Rules can also be used to represent standards and guidelines.
Action	
Definition	An Action is an activity that is performed in order to change a state of affairs. Actions correspond to the capabilities possessed by Agents.
Description	Agents, Rules and Resources can only interact with each other through Actions.
Examples	Actions may be: <ul style="list-style-type: none"> ▪ Addition, modification or deletion of data, information or systems

	<ul style="list-style-type: none"> ▪ Identification or selection of data, information or systems
<i>Notes</i>	<p>Many modelling languages identify the concept of an event that triggers some action. In fact, an event can simply be viewed as an action that is performed by another agent and modelled this way in LEAN.</p>

Table 6 - Definitions of Agent, Resource, Rule and Action Concepts

The pertinence of views to EA was discussed in Section 2.4.3. To recapitulate, a view is a model of a system, and while it addresses the whole system, it does so with respect to the concerns of a particular type of stakeholder (Hilliard, 1999a). The development of a full analysis of view and viewpoints, and their relation to the LEAN ontology is beyond the scope of this research. However, we can make some basic observations relating to the manner in which this area may be developed.

In the vernacular of LEAN, stakeholders are referred to as agents. Thus, views need to be based on an understanding of the Agent and their areas of concern. An Agent's area of concern can be evaluated based on the Actions that they can perform: a concern that does not have a potential source of action would be completely ineffectual and irrelevant. This provides a theoretical basis upon which viewpoints can be determined, based on the LEAN ontology, and commensurate views developed. These notions are illustrated in Figure 19.

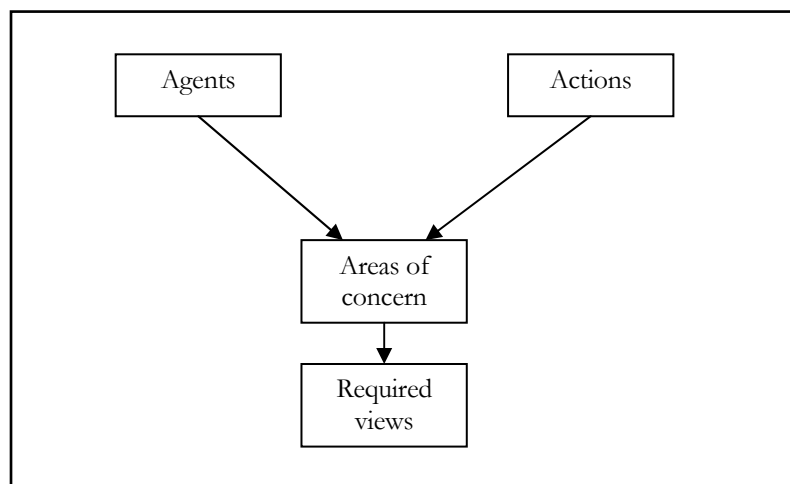


Figure 19 – Views, Viewpoints and Architectural Areas of Concern

The benefits of being able to *visualise* an EA and the role that graphical languages have in supporting visualisation were previously described (Section 2.4.4). It is desirable to develop an ontology that will lead to a graphical modelling language, consisting of nodes, connected by arcs (*all* graphs consist of nodes, connected by arcs). Since the nodes that we will be using represent concepts, and the arcs, relationships, the graphs are essentially ‘concept maps’ (refer to Section 2.4.4). The foundation of this graphical modelling language is described as the LEAN Topology.

5.3.2.2 *The LEAN Topology*

There is a wide variety of ways in which we could represent the four societal concepts using a graphical notation. The topological alternatives are then restricted to the following:

1. One or more of the concepts are represented as nodes. The remaining concepts are represented as relationships that connect these nodes.
2. The concepts are all represented as relationships that are used to connect other, predefined or user defined nodes.
3. All of the concepts are represented as nodes. A separate, predefined set of relationships is used to connect these nodes.
4. All of the concepts are represented as nodes. A separate set of relationships is used to connect these nodes, but these relationships are not predefined.

An example of the first case is to use the concept of Rule to define the relationships between Agents, Actions and Resources. For example, an Agent performs an Action according to some Rule, or an Action consumes a Resource according to some Rule. The drawback with this approach is the limited number of relationship types that will be possible (no more than three), limiting the semantic power of such a language.

In the second case, all of the concepts are represented as relationships and nodes are developed independently of these. However, it is hard to envisage the validity of representing an Agent or Resource as a relationship except in rare cases. These are typically thought of as entities, not relationships.

In the third case is where we define a fixed number of relationships that can be used in the model and four node types are used to represent each of the structural concepts. For example, we can define certain hierarchical relationships such as component-subcomponent and type-subtype relationships as being the only types of relationships permitted. All models would have to be built connecting the Action, Agent, Resource and Rule nodes using only the pre-defined relationship set.

The fourth case is similar to the third, except that the relationships are *not* predefined. The user defines any relationships that they need to model a given environment at a given time. This option clearly provides for the greatest expressive power. However, it does this at the expense of standardisation and, perhaps, formality (if the relationships are not well defined).

From a practical point of view, it could be argued that a combination of the third and fourth cases provides the most useful topology. That is, all four of the societal concepts are represented as nodes and a separate set of relationships is used to connect these nodes. These relationships come from two sources: a predefined set (providing an element of standardisation and formality), plus, optional user created relationships that are specific to a given environment or purpose (providing flexibility and relevance). This topology will now be developed further.

5.3.2.3 The LEAN Syntax

With the LEAN ontology captured, we can now more formally define the LEAN syntax, which fulfils the coding (formalisation and implementation) stage of the ontology construction.

We have stated that the four societal concepts (Agent, Action, Rule and Resource) will be represented as LEAN nodes. We can now associate graphical icons with each of these to provide visual support. These are shown in Figure 20.

A heuristic approach was used to develop these icons upon the following requirements. Firstly, the development of EA's takes place in a highly collaborative environment where models are often drawn by hand on paper or on whiteboards. Therefore, it is desirable to have icons that can be drawn by hand easily and quickly. Secondly, in order to make the icons as easy as possible to identify (even when they

may appear as small objects on a computer screen) it is desirable to make them each as distinct, and different as possible. The use of simple geometric shapes that are easily identifiable and distinguishable satisfies these criteria.

The most simple, distinct shapes are a circle, square and triangle. The triangle shape was chosen to represent the Rule concept, as we often associated this type of shape with street warning signs (i.e. road rules). The square has been modified by curving the horizontal lines. This was done to ensure that it would not be confused with a circle if drawn quickly using a single line (that tends to curve the corners). This gave the shape a 'dynamic' look, and so it was chosen to represent the Action concept. Since we commonly draw icons to represent people, we can easily come up with simple geometric figures, similar to 'stick figures'. Since a person is one type of Agent, the shape shown in Figure 20 provides an intuitive link to the Agent concept. This leaves the circle to be used to represent Resource.

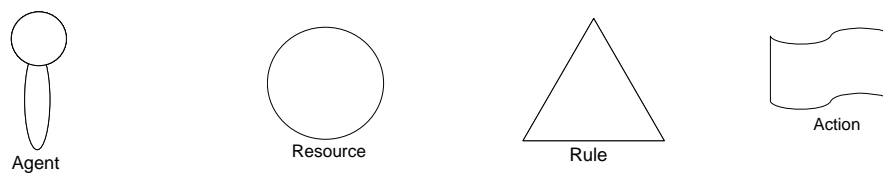


Figure 20 - The Graphical Representations of LEAN Nodes

As mentioned previously, the LEAN syntax consists of two structures: nodes and arcs. A LEAN graph contains one or more nodes, connected by zero or more arcs. An arc is connected to exactly two nodes, with one node attached to each end of the arc. A node is connected to zero or more arcs. However, each pair of nodes may only be connected by a single arc. Thus, LEAN models may be connected (where there is a path between every pair of nodes in the graph) or disconnected graphs.

A LEAN arc connects two nodes and is used to represent interdependency. Two connected nodes are called a pair. LEAN arcs have an arrow on one end that shows the direction of the relationship. Thus, LEAN graphs are directed, bipartite graphs.

The location of nodes within a graph is arbitrary. Similarly, the length, thickness or pattern of arcs is irrelevant.

LEAN arcs are drawn as smooth, continuous curves, rather than lines that change direction using sharp angles, such as right angles. Although links between graph nodes are often drawn using straight lines and sharp angles, it is far easier to perceive connections between nodes when the contours connect smoothly. (Ware, 2000, p.207)

The semantics of any individual pairing is indicated by a textual description associated with the arc connecting the two nodes. Thus, LEAN graphs are directed graphs. Reading in the direction of the arrow, these elements form a triple of the form subject-predicate-object. For example, the graph in Figure 21 is read “LEAN is a type of Modelling Language”.

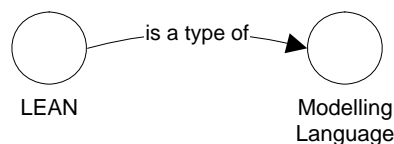


Figure 21 - A LEAN Relationship

There are three general categories of relationships that can be formed using LEAN:

- one-to-one relationships, such as husband and wife;
- one-to-many relationships, such as manager and employees; and
- many-to-many relationships, such as a product and all its parts.

Where these relationships are between occurrences of the same entity, they are termed "recursive relationships". (Haughey, 2005)

At this stage we have defined four types of LEAN nodes, described the types of relationships that can be formed between LEAN nodes (to form graphs), and described the way in which arcs are labelled. Note however, that we have not defined the semantics that these relationships represent. We will now build upon this foundation to produce a language that has the semantic breadth to be useful for real EA modelling situations.

The use of just four, highly generic, node types would make LEAN graphs extremely general. Even high-level EA modelling requires the identification of objects at lower levels of abstraction than the concepts of Resource, Rule, Action and Agent. We now

introduce a mechanism to introduce more granular nodes (nodes defined at a lower level of abstraction). To do this, we start by defining a set of LEAN relationships.

We term a collection of relationships a 'Relationship Set'. Therefore, a unique relationship set could be created for every EA project. However, since it is likely that certain 'generic' relationships are likely to be used frequently, even in different environments, these are provided as part of the LEAN syntax. These generic relationships are referred to as 'Reference Relationships'. Table 7 shows these Reference Relationships and indicates to which node pairings each type of relationship applies. In effect, this generic LEAN relationship set serves as a starting point for developing an enterprise-specific relationship set.

A starting point in defining useful LEAN relationships is the observation that we will clearly need to represent hierarchical relationships if we are to model enterprise systems. Hierarchies are a fundamental structure in these types of environments.

“A *hierarchy* is a set of variables which represent different levels of aggregation of the same dimension and which are linked between them by a mapping. A typical example of hierarchy is City → State → Region → Country.” (Pourabbas and Rafanelli, 1999)

There are many ways to represent hierarchical data, and the term 'hierarchy' can be taken to mean several things. Three types of hierarchy that are particularly relevant to organisational modelling are type-subtype hierarchies, reporting hierarchies, and component-subcomponent hierarchies (also referred to as a 'bill of materials' or BOM, or 'adjacency model', or 'parts explosions'). A component-subcomponent hierarchy is a many-to-many, recursive relationship. A component-subcomponent hierarchy can be used to represent, for example, a true BOM, organisation and employee structures, financial relationships and reporting rollup structures. (Haughey, 2005)

Thus, the Relationship Set has provision for the representation of three types of hierarchical relationship. These are:

- 'is a type of', which provides the semantics to represent type-subtype relationships.

- 'is a part of', which provides the semantics to represent component-subcomponent hierarchies.
- 'reports to', which provides the semantics to represent reporting hierarchies.

LEAN hierarchies are acyclic graphs. That is, a LEAN node can have more than one parent. With respect to type-subtype hierarchies, this infers that LEAN hierarchies can support multiple inheritance.

It will be observed that Agent, Action, Resource and Rule types are each minimal common supertypes of a type hierarchy¹³. Thus, there are four possible type hierarchies within any LEAN modelled enterprise. A type with no subtypes is called a 'base type'. The subtypes within each of these hierarchies inherit the attributes of its parent. Moving down through the hierarchy, the types become more specialised, detailed and domain specific.

Another type of relationship that is fundamental to the representation of enterprise systems is a temporal relationship, which can be used to represent process flows. This is represented in the Relationship Set using the 'precedes' relationship which provides a very commonly used temporal relationship. It may be desirable in future versions of the Relationship Set to include other types of temporal relationship such as 'overlaps', 'intersects', 'triggers', 'includes' etc. This depends on whether these relationships are found to be relevant and useful to EA modelling.

The remaining relationships in the Relationship Set have been identified and included through a heuristic process that included the following steps:

- Identify a relationship that cannot currently be described.
- Ensure that the relationship is unique and cannot be described using an existing relationship within the Relationship Set.
- Define the relationship.

Furthermore, it is observed that, in addition to hierarchies and processes, ontologies typically include information such as properties, value restrictions, disjointness

¹³ The minimal common supertype is also called the 'universal type' or 'root'.

statements and logical relationships (Antoniou and Harmelen, 2004, p.10). The remaining relationships that make up the Relationship Set fall within these categories. The entire Relationship Set is summarised in Table 7.

It will be noted that relationships in the Relationship Set fall into two sets: those between homogenous pairs of nodes and those between heterogeneous pairs of nodes. All possible pairings have been shown, except for the Agent-Resource pairing. This is because, at this stage, no useful *direct* relationships have been identified between an Agent and Resource (an Agent would typically use or produce a Resource by performing some Action upon it). However, if this relationship is needed in the future, an extra column can simply be added to the relationship table.

Note also that the semantics of the heterogeneous relationships are such that the arrow on the arc usually points away from the Action type. For instance, we say, “an Action is performed by an Agent”, rather than “an Agent performs an Action” (the one exception to this is the relationship “Rule applies to Action”). This custom makes it easier to remember how these graphs are drawn and read.

The Action-Rule, Agent-Rule and Resource-Rule relationships of ‘complies with’ and ‘applies to’ are semantically equivalent. The two models at the top of each of these columns indicate that the direction of the arrow can be changed so that a more natural terminology can be used to describe these relationships. The ‘complies with’ relationship is used when the arrow points towards the Rule, and vice versa for ‘applies to’.

As well as making it easier to start developing effective LEAN models, the generic LEAN Relationship Set also serves another purpose. The translation of LEAN models into domain specific architectural models requires that the semantics of each LEAN relationship is understood, and equivalence is found between that LEAN relationship, the objects it connects, and the domain specific notation to which we are translating. If we only know the node type, then it is impossible to develop a set of heuristics for translating a LEAN model into a meaningful domain specific model. However, if we have a predefined relationship set, plus a set of node types, then we *can* develop heuristics for translating a LEAN model into a meaningful domain specific model. Of

course, if an enterprise uses other relationships (either instead of, or in addition to the generic set) then the translation of these graphs into domain specific models cannot be predefined. However, the existing defined translations can serve as a guide, making the translation of new relationships a relatively easy exercise. In fact, in many cases the new relationship will be semantically close to an existing one, meaning that the existing translation can be used as a guide. This may happen, for instance, because an enterprise prefers to name relationships using the vernacular to which they are accustomed.

The four concepts of Resource, Rule, Action and Agent are termed ‘Universal’ types. ‘Non-Universal’ types can also be represented as nodes: homogenous LEAN pairings provide a mechanism for developing non-Universal types. For example, non-Universal types can be created for all four of the Universal types using the ‘is a type of’ and ‘is a part of’ relationships. However, the ‘supports’ and ‘interfaces with’ relationships can only be used to create non-Universal types of Resource, while the ‘reports to’ relationship can only be used to create non-Universal types of Agent (see Table 7).

When drawing non-Universal types, the label of Agent, Action, Rule or Resource is replaced with the name of the non-Universal type. For example, the graph in Figure 22 shows that the resource ‘Network Infrastructure’ is a non-Universal type of the Universal type ‘Resource’.

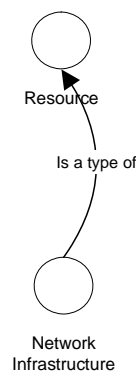


Figure 22 - A LEAN Universal and Non-Universal Type

LEAN NODE PAIRINGS									
RELATIONSHIP SET	HOMOGENOUS PAIRINGS				HETEROGENOUS PAIRINGS				
	 Action Action	 Agent Agent	 Resource Resource	 Rule Rule	 Action Agent	 Action Resource	 Action Rule	 Agent Rule	 Resource Rule
<i>is a type of</i>	✓	✓	✓	✓	✗	✗	✗	✗	✗
<i>supports</i>	✗	✗	✓	✗	✗	✗	✗	✗	✗
<i>interfaces with</i>	✗	✗	✓	✗	✗	✗	✗	✗	✗
<i>is a part of</i>	✓	✓	✓	✓	✗	✗	✗	✗	✗
<i>precedes</i>	✓	✗	✗	✗	✗	✗	✗	✗	✗
<i>reports to</i>	✗	✓	✗	✗	✗	✗	✗	✗	✗
<i>performed by</i>	✗	✗	✗	✗	✓	✗	✗	✗	✗
<i>uses</i>	✗	✗	✗	✗	✗	✓	✗	✗	✗
<i>produces</i>	✗	✗	✗	✗	✗	✓	✗	✗	✗
<i>complies with</i>	✗	✗	✗	✗	✗	✗	✓	✓	✓
<i>applies to</i>	✗	✗	✗	✗	✗	✗	✓	✓	✓
<i>supports goal</i>	✗	✗	✗	✗	✗	✗	✓	✗	✗

Table 7 - Mapping Between a Generic Relationship Set and the Range of Possible Node Pairings

5.3.3 Assumptions

In addition to the documentation already provided above, it is important to add documentation of all important assumptions such as assumptions “about the main concepts defined in the ontology, as well as the primitives used to express the definitions in the ontology ...” (Uschold et al., 1997)

One assumption that has been made is that Giddens's theory of structuration provides a valid theory for understanding societal structures. Giddens's work has received a great deal of review in the literature (refer (Poole and DeSanctis, 2003) (Clarke et al., 1990) (Bryant and Jary, 1991)). While not all reviews have been entirely favourable, on the whole, Giddens's work is well received.

Another assumption that underlies this work is that all organisations are composed of similar fundamental structures. That is, if one organisation's enterprise architecture can be modelled using a single ontology, then they all can. Clearly, we can continue to raise the level of abstraction of an ontology until this assumption becomes true. For example, if the ontology has a one to one correspondence with our working definition of an enterprise, then the language will be valid for all enterprises. However, in this case, use of the ontology adds little value: it can be used to describe a whole enterprise, but not any of the structures within an enterprise. Conversely, as the level of abstraction decreases, the ontology will become more specific, and may not be useful in describing a range of enterprises. The key is to find a balance between the two. The LEAN ontology is developed at a very high-level of abstraction as it is based on the societal metaphor, yet it is composed of a set of structures that are likely to be useful in describing a range of enterprises (because metaphors are hierarchical). Therefore, it is likely to be relevant to most, if not all, enterprises.

5.3.4 Limitations

In its current form, the ‘interfaces with’ relationship can be used to show that two resources interface with each other, but it does not show whether information flows in both directions, or just one direction. This limitation could easily be overcome by adding a new relationship. For example, a ‘sends information to’ relationship could be used to represent a unidirectional flow of information where the relationship arrow shows the direction of that information flow.

It is difficult to represent ‘negatives’ using LEAN. For example, it is difficult to show that a relationship to a resource, rule, action or agent does *not* exist. In some cases, it may be desirable to show, explicitly, that a structure does not exist.

5.4 The LEAN Modelling Tool

The LEAN modelling tool is designed to support the strategic planning of enterprise IT systems. The tool has been developed as an ‘add-on’ to Microsoft Visio, a graphical modelling tool, and has been created using a combination of stencil, template and ShapeSheet customisations.

Microsoft Visio is designed for the Windows platform and the LEAN modelling tool runs on Microsoft Visio 2002 and Microsoft Visio 2003.¹⁴ Visio is in use within most large IT departments worldwide: “... most organizations still use Microsoft’s office and Visio products for capturing their Enterprise Architecture results.” (Institute for Enterprise Architecture Developments, 2004)

Visio has a number of features that make it likely that it will continue to be a dominant EA modelling platform in the future. These include the ability to integrate with a wide range of data sources, collaboration facilities, support for business process management and programmability. Visio 2007 is currently in Beta testing so it is likely that Visio will be available and supported for some time to come.

Typically, business practitioners are reluctant to adopt *specialised* enterprise modelling tools and prefer to stick to the standard office tools and applications with which they are familiar:

In spite of all the trouble it takes to modify the standard office tool, they are frequently used for modeling. Because business practitioners are used to them, they know how to use them and ... these tools are already available and installed in most of the enterprises. There is no extra tool acquisition costs related to modeling if the model is built by a standard office tool.
(Szegheo, 2000, p28)

It can be inferred that a modelling language that can be deployed using standard office tools will have a higher likelihood of adoption. This makes Microsoft Visio a favourable platform for deploying the LEAN modelling tool.

¹⁴ It is expected that it will also run on later versions as they become available.

Version 1 of this tool included programs written using Visual Basic. These programs enforced the LEAN syntax and generated various derived views of the models that were developed by the user. For instance, illegal connections between objects are immediately identified: when an invalid connection is made, the connection is broken, the arc is moved away from the object, or objects, to which it has been connected, and the user is informed that an illegal connection has been made and is provided with information on the connections that can be legally made using that connector type. The views that can be created include process models, hierarchical models, and models that resolve multiple instances of a node into a single graph.

A significant effort went into developing this version of the tool¹⁵ as it was assumed that the syntax checking and automatic view generation features would be found useful by modellers. However, the initial feedback from Study One (described in chapter 7.1) showed that the users preferred a much simpler system with fewer constraints. Thus, the automation features were discarded and a much simpler Version 2 was put into production.

Features of the LEAN Version 2 modelling tool include:

- A customised stencil that provides LEAN icons and arcs for use in any drawing. (Feature implemented by creating a Visio stencil (.vss file) and including this in the Visio solutions directory.
- A customised Visio template that automatically presents a LEAN stencil and a new drawing page on Visio startup. (Feature implemented by creating a Visio template (.vst file) and including this in the Visio solutions directory.
- Document header and footer automatically produced when the document is saved. This includes the document's title, drawing path and filename and is displayed when the document is printed or print previewed. (Feature implemented using Visual Basic.)
- Icon text is automatically scaled as icon is resized. (Feature implemented by modifying icon ShapeSheets.)

¹⁵ Approximately six thousand lines of VBA code have been custom written to produce Version 1.

5.5 Summary

The LEAN ontology is relatively simple making it resilient to change and widely applicable. While most enterprises have fundamental structures that are similar, they still vary greatly in their form and purpose. It is possible to develop a far more detailed and prescriptive set of elements with which to describe an enterprise, however this would be committing to far more “ontological commitment”, as Gruber (1993) puts it, than is either, *necessary*, to describe an enterprise at a high-level, or *valid*, since there would be a greater likelihood that enterprises exist where these additional constraints do not hold. Also, the advantage of developing a simple ontology is that the provision of a small number of highly versatile, generic concepts upon which to develop a language can make it easier for users to remember the language, and can make problem solving quicker and easier. (Pawson, 2000, p.85)

The mindful use of metaphor as an aid to learning and cognition has been previously discussed as a powerful tool. New facts must fit into *pre-existing* concepts, “Otherwise facts go in and then they go right back out.” (Lakoff, 2004) Metaphors such as the societal metaphor of enterprise, allow us to place new information within an existing conceptual framework. Lakoff refers to this existing conceptual framework as a ‘frame’.

By applying an understanding of dynamic type hierarchies to the selection of system metaphors, it is possible to identify a metaphor that provides a scope that is wide enough to cover all of the functional elements of the enterprise’s component structures. After all, it is better to have a single metaphor that covers the entire domain, for “when discourse becomes full of conflicting metaphors, it may be difficult for the uninitiated to keep their bearings.” (Johnson, 1994)

This metaphor can then be used to develop a unified language that models all of the enterprise’s systems. Furthermore, the enterprise supertype based metaphor can be used to structure systems at various levels of abstraction and at different stages in the life cycle. This ensures consistency between different levels of system design. Since a single metaphor is used to replace a myriad of disparate, and possibly, contradictory metaphors, the resulting models have the following qualities:

- Greater explanatory power (a consistent metaphor/language is used to model the entire organisation, which allows the user to develop an effective mental map of the organisation).
- Greater flexibility for the management of change and strategic planning (current disjunctions between systems, business units etc. are not 'hard wired' into the model).
- Avoiding loss of information that might occur in translating from one architectural view to another and ensure cross-view consistency (Armour et al., 2003).
- Explicit capture and representation of business needs within the EA models (Armour et al., 2003).
- Reducing cognitive load placed on a user that needs to understand a complex set of architectural views (Armour et al., 2003).

The development of a language based on a highly conceptual metaphor naturally leads to the development of highly abstract models. One of the advantages of this is that it helps EA developers to remain at a highly abstract level when creating enterprise wide models. "We believe that the selection of the level of abstraction is especially difficult when no context is provided that helps one to remain at that level. Without a stable context, many people seem prone to drift from one level of abstraction to another." (Biemans et al., 2001) The EA modelling approach described herein avoids this problem by keeping the enterprise architect at an appropriate abstraction level through the provision of highly primitive semantic structures.

The use of a single metaphor to describe all enterprise systems helps break down the boundaries between applications, making them appear seamless and integrated. In essence, the enterprise systems become a single, integrated, system. The advantages that can accrue from having a single system, rather than multiple systems, are numerous and include reduced training and support costs, increased flexibility and positive organisational culture changes (Pawson, 2000, p.67).

The use of a very high-level EA language may actually supplant the need for an EA framework. Because the LEAN language describes the enterprise using a highly

abstract metaphor, there is no need to develop a generic framework within which the enterprise models must fit. In fact, the metaphor itself serves as the framework, but because of its inherent flexibility, it can be far less constraining than a traditional framework, allowing the structure of the organisation to be addressed on its own terms.

6 EXPERIMENTAL RESEARCH METHODOLOGY

6.1 Introduction

There are two, fundamentally different, research approaches in current widespread use: the quantitative approach, and the qualitative approach (Figure 23). This section provides a short review of these approaches, and introduces some of the relevant methodologies that are commonly used to gather and analyse research data. These approaches are then analysed with respect to the needs of the current research in order to identify the most suitable approach for answering the research hypothesis.

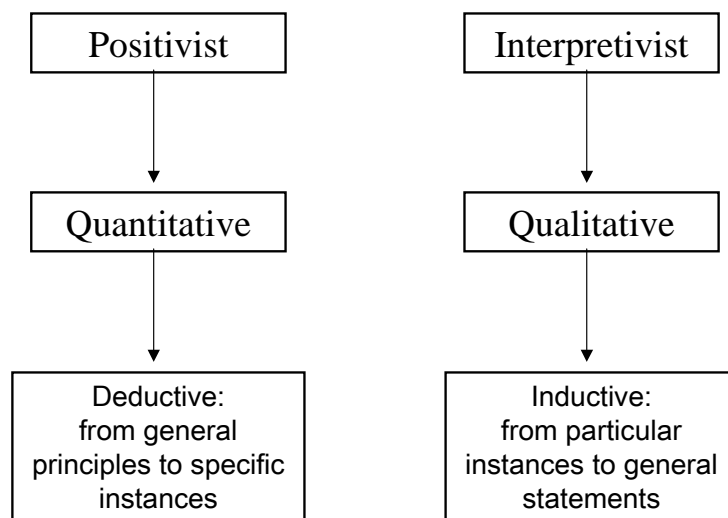


Figure 23 - Primary Research Approaches

6.2 Quantitative Research Methods

Of the two research approaches discussed here, the quantitative approach has the longest and best-established pedigree. Quantitative enquiry is used to test specific hypotheses that are usually part of a broader theoretical perspective. The approach fits naturally with the positivist (or objectivist) paradigm, emphasising standards, precision and reliability (Slembek, 2003). It takes a scientific, empirical, approach that focuses on highly reproducible data collection and analysis.

However, despite its favoured position, the positivist paradigm is not without detractors. One basis for criticism is that it has limited potential for dealing with

complex data and interactive phenomena in dynamic, real-life environments. According to Guba and Lincoln, “Phenomena can be understood only within the context in which they are studied; findings from one context cannot be generalised to another; neither problems nor their solutions can be generalised from one setting to another.” (1989, p.45) Quantitative research methods attempt to create experimental environments that negate any influence that arises from the contextual setting of the study. However, it is clear that in many cases there can be, at best, only limited control of the experimental setting.

Methodologies within quantitative research include experimentation, deduction and formal survey methods. Evaluation of this research using quantitative methods entails the development of both a hypothesis and null hypothesis. In this case, they are as follows:

- Hypothesis: It is possible to develop a human-centred modelling language for creating unified models that span heterogeneous domains of an enterprise architecture.
- Null hypothesis: LEAN is not a human-centred modelling language for creating unified models that span heterogeneous domains of an enterprise architecture.

6.3 Qualitative Research Methods

Qualitative methods emphasise the richness of description in data collection, focussing not just on outcomes, but also the social processes in an organisation (Slembek, 2003). Qualitative methods are based on the interpretivist (or subjectivist) paradigm, which relies on the experience and background of the evaluator. The tacit knowledge that the evaluator brings to the research is likely to affect their perceptions of the observed phenomena, and thus, their research conclusions. The main criticism of the interpretivist approach concerns the potential variability of research outcomes and the inherent inability to replicate results (Worthen et al., 2003).

There is a wide variety of qualitative research methods. However, some of the more well established, and widely used approaches, include action research, case study research, ethnography and grounded theory. These methods are discussed in more detail below.

6.3.1 Action Research

Action research is an interpretive approach "concerned with the study of human actions and social practice." (Williamson, 2002, p.159). In action research, the researcher is an active participant rather than an independent observer (Williamson, 2002, p112).

Action research *is* aimed at informing theory and creating knowledge. However, it can also be used to bring about an improved practice or propose new solutions to immediate and practical problems (Williamson, 2002, p.161) (Baskerville, 1999). This provides a significant benefit where the research needs to be undertaken within a corporate setting, and where permission to conduct the study may only be forthcoming if there are likely to be ensuing benefits for the participants. In a corporate setting, the cost of resource allocation is keenly felt and any study that requires time and effort from corporate staff will be highly scrutinised.

One weakness of action research is that, like case studies, action research is usually concerned with single situations such as a particular enterprise. "Therefore, although the approach can generate theoretical propositions that go beyond single situations, action research is seldom seen as an appropriate approach to test the general applicability of theories." (Williamson, 2002, p.161)

While we are seeking to test a well-developed and specific hypothesis for its applicability to a wide domain, the use of action research, by itself, will not be complete and sufficient. However, action research is likely to provide some invaluable, qualitative data that could not be obtained using other methods.

6.3.2 Case Study Research

Case study research is the most common of the qualitative methods that are used in information systems research (Myers, 1997). In fact, it is particularly well-suited to IS research as it supports the shifting focus from technical issues that lack context, towards an understanding of the organisational issues that concern information systems (Myers, 1997) (Williamson, 2002, p112).

In case study research, the researchers do not control the program in any way. They merely observe it in order to examine what may be a wide range of intended and unexpected outcomes (Stufflebeam, 2001). In contrast to action research, “the researcher usually has little or no capability of manipulating events ...” (McCutcheon and Meredith, 1993). In fact, the observed case may even have occurred in the past.

Case study data “may come from primary sources (such as direct observation or interviews of people involved) or secondary sources (documents or records, for example). It may examine a single situation or, with multiple-case studies, several related situations.” (McCutcheon and Meredith, 1993)

However, case study research lacks relevance where a specific hypothesis has already been developed, or where an understanding of the mechanisms that *cause* a phenomenon to occur, are not of interest. As we already formulated the research hypothesis under evaluation, case study appears to offer little benefit.

6.3.3 Ethnographic Research

"Ethnographic research comes from the discipline of social and cultural anthropology where an ethnographer is required to spend a significant amount of time in the field. Ethnographers immerse themselves in the lives of the people they study ... and seek to place the phenomena studied in their social and cultural context." (Myers, 1997)

According to Cavaye (1996) as referenced in Williamson (2002, p.112), "Ethnographic research differs from case study research in that the findings are not usually related to generalisable theory and are interpreted from the researcher's point of view."

6.3.4 Grounded Theory Research

"Grounded theory is a research method that seeks to develop theory that is grounded in data systematically gathered and analyzed. ... The major difference between grounded theory and other methods is its specific approach to theory development - grounded theory suggests that there should be a continuous interplay between data collection and analysis." (Myers, 1997)

In the case of this research project, there is a *specific* theory that requires testing. This makes grounded theory research a less suitable approach for achieving the research goals.

6.4 Chosen Methods

The choice of a research method should be based on two criteria: the type of question that is being asked, and the audience for the research (Williamson, 2002, p35) (Stufflebeam, 2001).

As the evaluation of LEAN (as a user friendly, unified modelling language) is essentially the testing of an *invention*, it is desirable to avoid an interventionist approach where possible. An inventor/researcher can be expected to have strong a priori beliefs regarding alternative EA approaches and a personal prejudice for the invention, so an interventionist approach may lead to skewed results. Furthermore, it would be attractive, in terms of accuracy and repeatability, to be able to evaluate this research using purely quantitative methods.

Unfortunately, the use of purely objective research methods is unlikely to be practical in this case. In order to assess the validity of the hypothesis it is desirable to conduct, at least some of the research, within a realistic setting where the tool in question is applied to a real problem by the practitioners for whom it is designed. Clearly, "... investigating ongoing business operations does not allow conditions to be controlled or variables to be manipulated ..." (McCutcheon and Meredith, 1993). Thus, an effective assessment of this research will require at least some use of qualitative research methods.

Fortunately, quantitative and qualitative research approaches can complement each other, resulting in a method that is standardised and reliable, yet also retains some depth that makes the results interesting and relevant to the target audience. The combination of methods can also cross-validate the findings (Stufflebeam, 2001). In fact, in combination, these different approaches can provide a large scale picture while simultaneously providing a more detailed understanding of a specific situation (Williamson, 2002, p.35). Furthermore, there is a large, well-established practice of

mixed methods research. Stufflebeam (2001) even goes as far as to say, “It is almost always appropriate to consider using a mixed-methods approach.”

In this case, it will be clear from the above discussions that the most favourable approach to the *qualitative* analysis of this research is to use action research. Therefore, the method chosen to evaluate the hypothesis presented in this thesis is to use a combination of experimental research and action research.

Consequently, a combination of three significantly different approaches will be used to evaluate the hypothesis. These approaches are:

1. An action research program that applies the use of LEAN to develop a set of EA models for a real enterprise IT department. Relevant personnel from that enterprise, as well as independent enterprise architects, will then evaluate these models, and the overall effectiveness of LEAN.
2. The remodelling of a large scale, existing EA, using LEAN. A comparative analysis is then performed between the original EA models and the new LEAN models.
3. The application of a Game based Ontology as a mechanism for improving the structure of lower level architectures. In this case, the concept of a Game is used as a unifying, highly conceptual metaphor and a simple Ontology is developed around this concept. Then, a set of commercially available email interfaces is reconstructed using this Game Ontology. The restructured interfaces are then compared against the original interfaces.

The first two approaches are described respectively in the following chapters: Study One: Modelling a Large Enterprise and Study Two: Re-Modelling a Public Domain Architecture. The third approach is described in Appendix F – Designing and Re-Engineering Subsystems, as it falls outside the formal scope of this thesis and is provided primarily as a basis for ongoing research.

6.5 Summary

There exists a wide variety of potential research methods: each with their own strengths and also their applicability in terms of the question being asked and the audience to which the research is to be supplied.

In this case, the use of complementary research methods - experimental research and action research - has been identified as providing the greatest potential for evaluating the research hypothesis.

7 EXPERIMENTAL STUDIES

In this chapter, two separate implementations of the LEAN language are presented. The results of these studies are analysed and the results are presented, along with an evaluation of the language and supporting theory.

7.1 Study One: Modelling a Large Enterprise

7.1.1 Introduction

Study one entails an action research program that applies the use of LEAN to develop a set of EA models for a real enterprise IT department. Relevant personnel from that enterprise, as well as independent enterprise architects, then evaluate these models, and the overall effectiveness of LEAN.

The development of enterprise architectures provides enterprises with strategic competitive value. Therefore, the information contained within the enterprise architectures of commercially oriented companies is usually considered highly confidential. This can make research in this area difficult. However, public enterprises such as government and teaching institutions that operate in less competitive environments may be less sensitive about their IT plans and structures and potentially more amenable to the exposure of this information for research purposes. It was largely for this reason that the Information Technology Department (ITD) of the University of Technology, Sydney (UTS) was targeted for this research study.

ITD provides computing resources and consulting to a population consisting of 2,635 staff, and 28,256 students, both within the city campus and in distributed computing laboratories located at various campuses around metropolitan Sydney. Approximately two hundred and eleven major systems are managed, operated and supported, along with ongoing IT development activities.

This research project was conducted between the 10th August 2005 and the 14th October 2005. However, project scoping and definition took place over several weeks preceding these dates, and the initial contact and meetings with the CIO, in order to garner support for the project, took place in the first quarter of 2005.

7.1.2 Research Approach

The nature of this research demands that survey samples are very carefully selected. Typically, an organisation has only a few (if any) staff working at the enterprise planning level, while enterprise architecture in particular, is a highly specialised function that is typically performed by very experienced personnel who already have many years of IT experience. In addition, the nature of an EA project means that a relatively long time frame is required to gain the client's confidence that the project is worthwhile (since they need to commit significant resources to the project) and then to scope, execute and finalise the project. This further emphasises the need to carefully develop the research approach and to identify appropriate research targets.

In the work presented here, nine ITD personnel were identified as having functional responsibility relevant to this research. Two of these respondents are charged with EA responsibilities. The survey could have been extended to additional staff members in order to increase the sample size. However, this would have introduced a bias since the additional personnel involved cannot be assumed to have the relevant skills, background or responsibilities to assess the developed models.

In order to increase the sample size of enterprise architects, two approaches were considered:

1. Extend the research to multiple organisations.
2. Introduce independent enterprise architects into the survey.

While both options would have provided additional enterprise architect respondents, the first option would have required extremely long time scales to achieve a moderate increase in EA subjects, since most organisations employ very few (if any) EA's. However, option two allowed the specific targeting of EA's from any number of different enterprises, allowing us to increase this sample size easily. In addition, option two was found to be a preferable approach since it introduces *independent* analysts into the research. This is especially valuable in an action research based project where the business users may have a personal interest in the project outcome.

7.1.3 Project Outline

The scope and goals of this project are contained in the report “Project Summary for UTS Enterprise Architecture Project” that was submitted to UTS senior management for approval before work commenced on the actual EA modelling. A copy of this report is provided in Appendix A – Project Summary for UTS EA Project. This report provides an outline of the methodology, project plan and objectives for this project. The objectives are particularly important in understanding the relevance of this project to the research goals, and are reproduced in Table 8.

Project Objectives	
1	To produce a high-level view of the university’s Enterprise Architecture.
2	To show the interrelationships between the different domain architectures.
3	To describe the primary relationships between the targeted systems.
4	To identify the major infrastructure components that support these systems and show the linkages.
5	To identify the major business processes that are supported by the identified application and infrastructure components and show the linkages between them.
Customer Objectives	
1	To produce concise, easily understood, graphical models of the high-level Enterprise Architecture.
2	To develop models that show the interrelationships between business goals and objectives, and IT systems and services.
3	To use the Enterprise Architecture models to identify the impact of change.
Provider Objectives	
1	To create unified Enterprise Architecture models that span heterogeneous ICT domains.
2	To develop an Enterprise Architecture that is concise, easy to expand and modify, and easy to understand.
3	To develop an Enterprise Architecture that is effective as an enterprise planning and evaluation tool.

Table 8 - ITD Project Objectives

The modelling of a single system, UTSONline, was undertaken as a pilot study and proof-of-concept for ITD. Following the completion of this model, a report was produced and approval was gained to continue and complete the project with the modelling of an additional nine systems. These systems were selected (by ITD) on the basis that they are core university business systems with which ITD are heavily involved.

The LEAN models were produced using information obtained from documents that ITD had previously produced and from face-to-face interviews with stakeholders. These stakeholders possess specialist information about various ITD managed systems, and also have different perspectives of these systems. For instance, the stakeholders included (but were not limited to) the Network Manager, Applications Manager and Implementation Manager¹⁶. Meetings with these stakeholders were conducted in a collaborative manner. The goal was to get these stakeholders using LEAN in order that they could start to develop an opinion on its learnability, ease of use and efficacy. No formal training in LEAN was provided: stakeholders were merely told what the node and relationship symbols represented and walked through some draft models that related to their area of expertise. They were then encouraged to modify and augment the draft models according to their knowledge of the systems being modelled, which they did either on paper or using a whiteboard.

Once the project was complete, a final report was developed and presented to the ITIO group, which consisted of most of the stakeholders that were involved in development models, as well as other senior management. A copy of this report, including the fifteen LEAN models that were produced, is provided in Appendix B – Final Project Report for UTS EA Project.

7.1.4 The Survey Questions

The first step taken in developing the survey questions was to identify the specific research questions that comprise the hypothesis, and the specific observable qualities to be evaluated. These were shown previously in Table 1. These research questions and observable qualities were then used to develop some high-level survey questions that address these specific question and qualities, as shown in Table 9. This provided a logical framework for the specific questions that would be used to evaluate the research hypothesis: *It is possible to develop a human-centred modelling language that can be used to create unified models that span heterogeneous domains of an enterprise architecture.*

¹⁶ Access to these resources was provided by Ian Waters (Senior IT Program Consultant) who, along with Peter Demou (Manager, Plans and Programs), championed the project.

RESEARCH QUESTIONS	OBSERVABLE QUALITIES	HIGH-LEVEL SURVEY QUESTIONS	RESEARCH QUESTION REF.
Is LEAN human-centred?	Learnability Useability Understandability	How easy is LEAN to learn and use? Would subjects use LEAN again? Would subjects recommend LEAN to others?	A
Is LEAN unified?	Effectiveness Relevance	Is LEAN effective for modelling high-level information? What about low-level information? Can LEAN be used to capture information from different business and technical domains? Are LEAN models meaningful and do they augment human cognitive powers?	B

Table 9 - Study One Test Areas and Research Approaches

There are two separate sets of respondents for the evaluation, each with their own set of questions. The first set of respondents were the *business* users that were involved in the LEAN project, either by being involved in the modelling itself, or because they were a stakeholder in the results of the project. These employees of ITD included both senior technical architects as well as IT managers. This set of business users did include two subjects (the IT Plans and Programs Manager and the Senior IT Programs Consultant) who have direct responsibilities for enterprise architecture.

The second set of respondents consisted entirely of subjects who identified as having past experience with enterprise architecture, or current responsibilities that include enterprise architecture. These subjects were not directly involved in the business project and they have no relationship to ITD. This set of subjects evaluated the results of the business project and made an assessment of LEAN based on these results. Thus, these respondents provide an entirely independent view of the project's results, while also increasing the sample size of respondents who have experience with EA and understand the goals and challenges associated with EA¹⁷.

¹⁷ Even the largest of enterprises would typically employ less than a handful of EA's, if any. Therefore, in order to obtain a reasonable sample size of EA's for any one study, it is necessary to extend the survey beyond the enterprise.

With these two user groups in mind, the high-level survey questions were transformed into a set of highly specific questions that would be used to create two surveys: one for each user group. Most of the questions in the business user and EA surveys are identical. Question 1 (refer Table 10 and Table 12) differs in order to account for the fact that one group was developing with the LEAN language, while the other was just analysing it. Question 18 - “What other languages have you used for enterprise modelling?”, is unique to the EA survey.

The questionnaires applied in the study are shown in Appendix C – Questionnaires. They are each broken into two sections: closed questions and open questions. (The reference columns and question numbering system should be ignored as these have changed since the questionnaire was initially deployed.)

All of the surveys were conducted after the project had been completed. The surveys were emailed to the nine ITD *business users* on 10th October 2005. These business users were selected on the basis that:

- They all perform roles that are critical to the success of the IT department and that are tied closely to the business objectives of UTS.
- They all represent typical stakeholders in an EA program.
- They are all seasoned IT professionals with a wide variety of experience and exposure to IT planning and strategy methodologies and approaches.

All of these survey recipients replied to the survey by 21st October 2005.

The *enterprise architect* surveys were completed over a span of several months following project completion.

7.1.5 Results

The following sections present the survey results from the two different sets of respondents: Business Users (Section 7.1.5.1) and Enterprise Architects (Section 7.1.5.2).

The ‘Weighted Average’ column shows the weighted average for all responses *excluding* ‘0-don’t know’ responses. An arrow (→) is used to show the closest explicit response to that weighted average.

Responses to the open questions are provided verbatim (including grammatical errors).

A single dash (“-“) means that no answer was provided to an open question.

7.1.5.1 Business Users Survey Results

The following tables (Table 10, Table 11) summarise the survey results from business users.

RESULTS OF LEAN SURVEY (Business Users)										
Research Question Ref.	Question Number	QUESTION	0-don't know	1-very strongly agree	2-strongly agree	3-agree	4-disagree	5-strongly disagree	6-very strongly disagree	WEIGHTED AVERAGE
			TOTAL RESPONSES							
A	1	I found LEAN easy to use.	3		2	4				2.67 → Agree
B	2	LEAN is an effective language for modelling high-level (conceptual) information.	1		1	7				2.88 → Agree
B	3	LEAN is an effective language for modelling low-level (detailed) information.	4			3	2			3.4 → Agree
B	4	LEAN captures information across all technical domains of interest.	2		2	6				2.71 → Agree
B	5	LEAN captures information from all business areas of interest.	2		1	5				2.83 → Agree
B	6	LEAN leads users to think more deeply about the structures and relationships that exist.	1		4	4				2.5 – Agree / Strongly Agree
B	7	LEAN models convey more meaning than the models I previously used.	4	1		3	1			3.25 → Agree

B	8	LEAN models convey meaning more precisely than the models I previously used.	5	1		2	1			3.33 → Agree
A	9	I would use LEAN again for enterprise architecture modelling.	3		2	4				2.67 → Agree
A	10	I would recommend LEAN for use by other enterprise architects.	5		2	2				2.5 – Agree / Strongly Agree

Table 10 - Results of LEAN Survey for Business Users - Closed Questions

RESULTS OF LEAN SURVEY (Business Users)	
11	What is your job title?
11.1	IT Plans & Programs Manager
11.2	Senior IT Programs Consultant
11.3	Network Manager
11.4	Communications Systems Planner
11.5	Senior Business Analyst
11.6	Applications Project Manager
11.7	Director IT Infrastructure & Operations
11.8	Technical Implementation Manager
11.9	Information Systems Manager
12	What is your job function?
12.1	I supervise managers of 4 different functions: IT Purchasing Office, Network Operations Center, IT Security Office, and Project Management Office
12.2	Manager, IT Security Office and IT Architecture
12.3	Network Operations
12.4	-
12.5	Business Analysis, Project Management
12.6	Managing Integration and Training. Basically software development
12.7	As above
12.8	Implementation of new systems infrastructure and management of central computer operations
12.9	Manage teams delivering database administration for corporate Databases, team delivery administration/support of the e-learning environment team, delivering business intelligence admin/support, manages project manager delivery, major software selection/implementations
13	Based on your experience with LEAN, what is your opinion on its value as an enterprise architecture modelling tool?
13.1	I think it is an efficient and easy to understand way of capturing and conveying the relationships between

	components of the architecture. When included as a means of elaborating on the higher level views of the architecture it proves very effective for drilling down and I thought it was excellent for depicting the relationships in the information architecture domains because it is quite efficient at showing the relationships between info flows, agents, systems, and databases.
13.2	Very valuable – see Project Results in Project Report
13.3	Gives a clearer understanding of the interconnections of resources to provide services
13.4	-
13.5	I really have no experience with LEAN other than seeing the results presented. I also have no other experience with enterprise architecture modelling tools making it not possible to compare LEAN for its ease of use or effectiveness. The diagrams were easy to read, but I can't comment on how easy it is to use
13.6	It shows the interaction well enough, but still leads top complex diagrams, which negates its effectiveness. Possibly it is just the nature of the task not the tool.
13.7	Only indirect experience at present, but on the surface the value as a high-level modelling tools looks very promising
13.8	It has the potential to be quite valuable as long as all staff have the required understanding of the notation and how to read the diagrams.
13.9	I have had limited exposure to LEAN. As I was not the main agent developing the models, I cannot comment on how easy it is to maintain the models. From a user perspective LEAN is easy to use/understand, in particular at a high-level.
14	What do you see as the strengths of LEAN?
14.1	Simple to use Simple to understand what is depicted One picture is worth 1000 words Easy to read/not overly complicated
14.2	See Project Results
14.3	Simplicity
14.4	-
14.5	-
14.6	Well thought out legends with the ability to easily jump between different areas Of the industry
14.7	Easy of use, simple high-level EA tool
14.8	It enables high-level architectures to be visualised using simple components and rules.
14.9	Its graphic presentation is easy to follow
15	Do you have any suggestions for improving LEAN?
15.1	Not just now but perhaps as we learn more during the documentation of our architecture we may want some added features.
15.2	Not at the moment – other than continue to keep it simple and do not add complication
15.3	no
15.4	-
15.5	-
15.6	Not really

15.7	Not enough experience with the tool to have any suggestions to date
15.8	Clarity could be improved by colour coding the LEAN node types – red, blue, green, yellow.
15.9	I would suggest maintaining in parallel to the LEAN graphics or linked to it a database or alternative for storing the relationships. This approach would facilitate searching for a particular component, agent, rule etc... Also a legend of the meaning of notation would ensure that even others not familiar with LEAN could interpret it.
16	What do you see as possible areas for the further development of LEAN?
16.1	Can't comment now. Maybe something in the area of an added symbol that related more to the relationships in the infrastructure layers
16.2	See Project Recommendations
16.3	-----
16.4	-
16.5	-
16.6	-
16.7	N/A
16.8	An automated tool that enabled drill down or expansion of sections of a view or the entire view, much like a CASE type tool, would be good but is not strictly part of the methodology.
16.9	See Q5. Also developing a referencing standard to link the various drawings could be recommended
17	Are there any other comments you would like to make?
17.1	I was very impressed with the number of systems that were able to be depicted in such a short amount of time.
17.2	It was a pleasure working with you Gerald. You managed to achieve in a short period something that was required within ITD, and in such a way that it can be relatively easily expanded and extended should resourcing be approved. Thanks again for your assistance.
17.3	-----
17.4	I didn't have enough involvement in the use of LEAN and the development of the results to complete this questionnaire with any conviction.
17.5	-
17.6	-
17.7	Hoping the this will help gain the buy in from the management team to enable the ongoing documentation & maintenance of our IT architecture
17.8	I found the process very effective and time-efficient.
17.9	A very good start, I could see LEAN developing and improving given user feedback of those developing the models as well as those utilising the architecture models

Table 11 - Results of LEAN Survey for Business Users - Open Questions

7.1.5.2 Enterprise Architect Survey Results

The following tables (Table 12, Table 13) summarise the survey results from Enterprise Architects.

RESULTS OF LEAN SURVEY (Enterprise Architects)										
Research Question Ref.	Question Number	QUESTION	0-don't know	1-extremely easy to learn	2-very easy to learn	3-easy to learn	4-difficult to learn	5-very difficult to learn	6-extremely difficult to learn	WEIGHTED AVERAGE
			TOTAL RESPONSES							
A	1	Compared to other modelling languages I have used for enterprise modelling, LEAN is:		1	2	1				2 → Very easy to learn

Research Question Ref.	Question Number	QUESTION	0-don't know	1-very strongly agree	2-strongly agree	3-agree	4-disagree	5-strongly disagree	6-very strongly disagree	WEIGHTED AVERAGE
			TOTAL RESPONSES							
A	2	LEAN is an effective language for modelling high-level (conceptual) information.		1		3				2.5 – Agree / Strongly Agree
B	3	LEAN is an effective language for modelling low-level (detailed) information.	2				2			4 → Disagree
B	4	LEAN captures information across all technical domains of interest.			1	3				2.75 → Agree

B	5	LEAN captures information from all business areas of interest.			1	3				2.75 → Agree
B	6	LEAN leads architects to think more deeply about the structures and relationships that exist.			2	2				2.5 – Agree / Strongly Agree
B	7	LEAN models convey more meaning than the models I previously used to describe enterprise architectures.	1		1	2				2.67 → Agree
B	8	LEAN models convey meaning more precisely than the enterprise architecture models I previously used.	2		1	1				2.5 – Agree / Strongly Agree
A	9	I would use LEAN again for enterprise architecture modelling.	1		1	2				2.67 → Agree
A	10	I would recommend LEAN for use by enterprise architects.			1	3				2.75 → Agree

Table 12 - Results of LEAN Survey for Enterprise Architects - Closed Questions

RESULTS OF LEAN SURVEY (Enterprise Architects)	
11	What is your job title?
11.1	Principal Consulting Architect
11.2	Software Consultant
11.3	NSW RTA eBus Team Technical Mgr.
11.4	Integration Architecture Manager
11.5	
11.6	
12	What is your job function?
12.1	Designing and Implementing Solution Architectures
12.2	Analyst, Technical Architect, Software Designer
12.3	eBusiness/Internet applications manager. Review all applications technical architecture. Lead the team for using new technologies or architecture. Managing operations and infrastructure for all RTA's eBus applications (25 applications) Providing input to RTA's strategic technical direction.
12.4	Managing Solution Architects Unit (6 people)
12.5	
12.6	
18	What other languages have you used for enterprise modelling?

18.1	UML, BPEL (Business Process Engineering Language), OWL (Ontology Web Language)
18.2	UML / Custom notation that I know would be easily understood by CTOs / CIOs
18.3	Visio diagram MS Word based (home-grown) diagrams UML models
18.4	BPEL, UML
18.5	
18.6	
13	Based on your experience with LEAN, what is your opinion on its value as an enterprise architecture modelling tool?
13.1	Useful at the highest level to show how infrastructure and personnel support business processes. This would help at the inception level to measure the scope of proposed changes.
13.2	Excellent. LEAN provides a simple mechanism for describing enterprise architecture without introducing the complexity inherent in languages which operate at a lower level of abstraction. It effectively ties business functions, processes and systems together into a neat, easily understandable language. It offers great potential to help CTOs plan enterprise architecture, explain the business goals of technology initiatives to an executive audience, track configuration of existing architecture components and track project progress.
13.3	RTA has an Enterprise Architecture Group. Which has been struggling to define RTA's current Enterprise Architecture. And looks low as opposed to high most of the time. Not very effective in general. Lean like any other (effective tool) will be effective in the hands of effective people. An Enterprise Architecture team with no real power can't make a dent in short term views and plans. There is definitely value in Lean, it has some shortcomings but they are not major and it can be improved like any other tool.
13.4	In its early stages it potentially offers unification of the disparate notations used today for Enterprise Architecture model drawings. Simplified nodes type set would-be much easier to remember and implement in the existing toolsets.
13.5	
13.6	
14	What do you see as the strengths of LEAN?
14.1	Simple to learn Enables a diagrammatic representation of almost any interaction The links are explained in writing on the diagram rather than implicitly by the style of line or arrowhead. This means that almost no learning is required.
14.2	Easily understandable by a management executive audience Extremely easy to learn Effectively articulates business functions, processes, technology and relationships in a single, easily-readable language Potential to integrate with RUP artefacts, allowing top-down management and tracking of IT projects
14.3	-Its simplicity -Tries to map the real world as opposed to subjective names or entities -Tries to stay high -Generic in nature

	-defines relationships (which can be easily enforced if it is supported by a drawing tool) -Is extendible as an item can have children, so further drilling down is possible
14.4	Simplified notation, limited attributes on each node, easier to understand relationships. Potential to be easier to understand by not previously exposed groups (management, business users, etc)
14.5	
14.6	
15	Do you have any suggestions for improving LEAN?
15.1	As a provider of technical solutions I would find it more helpful to have the resource symbol a little less generic. The description helps but maybe there could be some kind of differentiation between for example networks and applications.
15.2	<ol style="list-style-type: none"> 1. Integration with RUP 2. Using LEAN to help tracking project status by management execs / steering committees etc.
15.3	<p>Yes;</p> <ul style="list-style-type: none"> - I wish relationships were all objective, i.e. A causes B and not D was the caused by C . e.g., A client performs an action, and not an action is performed by a client. This may not be possible, but if it could, then learning of LEAN becomes easier. - Would be good if what-if scenarios can be performed on LEAN diagrams, e.g., what if I remove this agent. (With an automated tool for LEAN this should be easy) [I know relationship “dependency” is defined] – but what if you need to have two relationship [a- is part of , b- is dependent on] - LEAN is to help with CHANGE. If actions, rules, agents, etc. had properties associated with them that would show say their suitability, age, lifecycle, etc. then LEAN could tell you, okay after 5 years you need to change this and that. And if you do this and that you also need to change those other two linked systems. - LEAN should help architects identify areas that are weak and need changes based on say security, usage, fault tolerance, response-time, etc. <p>Bottom line for change is \$, could lean associate cost of operations with actions or resources. Enterprise architects based on subjective and objective cost/benefit analysis should bring about change. E.g., if Window is costing so much then lets use LINUX. And if we do this, how we will save money, how many systems, subsystems, agents, actions will be affected and in which way.</p>
15.4	Question: Are the relationships/connections directed only on way? Improve definitions (maybe more examples) for nodes i.e. in common language agent and resource could be the same person, so we would need more precise definition.
15.5	
15.6	
16	What do you see as possible areas for the further development of LEAN?
16.1	Perhaps some implementation of Object Oriented principles (containments, aggregation, inheritance etc) Perhaps a grouping notation so that the enterprise can model for example what is in and what is out (of a proposed change) in one diagram
16.2	<ol style="list-style-type: none"> 3. Development and publication of an open LEAN modelling standard 4. Licensing of a LEAN modelling tool 5. Delivery of LEAN modelling education services to CTO / Enterprise Architects & Management Executives.

16.3	Please see above.
16.4	Align or link it to other notations like BPEL etc.
16.5	
16.6	
17	Are there any other comments you would like to make?
17.1	Would be very helpful to facilitate change in an environment where complex business processes are supported by a plethora of applications and rules.
17.2	-
17.3	There is definitely a need for such tools such as LEAN. If I were managing RTA's whole EA, I would definitely use something link LEAN. Hence, I don't see this as a waste of time. LEAN should not become another modelling tool. It should assist change in EA in a real sense. I.e., looking at costs, impacts, improvements, risks, and real things that help to run a business.
17.4	My exposure to the concept was very brief, and only daily usage would test the concept and usability and validity of the idea.
17.5	
17.6	

Table 13 - Results of LEAN Survey for Enterprise Architects - Open Questions

7.1.6 Analysis of Results

This section provides a preliminary analysis of the results of Study One, while Section 7.3 provides a comprehensive evaluation of the research.

7.1.6.1 Closed Questions

Figure 24 shows spider chart representations of the weighted average responses to the closed questions for each of the target groups. Actually, the *inverse* of the weighted averages are used so that values further away from the origin represent stronger agreement with the statement or question.

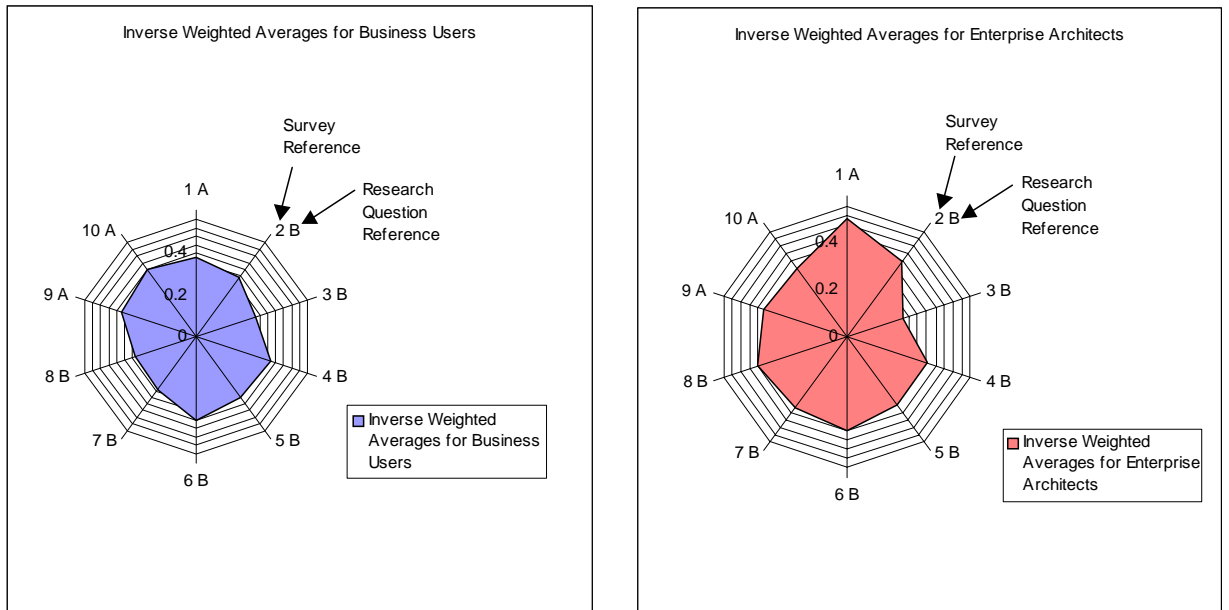


Figure 24 - Responses for Two Groups (Business Users and EA's) Compared by Question and Test Area.

The spider graph for *business users* does not reveal any striking trends with respect to the two research questions. However, the *enterprise architects* graph highlights a very strong response to question 1: “Compared to other modelling languages I have used for enterprise modelling, LEAN is:” which received a weighted average of “2 → Very easy to learn”. Both graphs show a local minimum for question 3 indicating that, while LEAN is seen as having *general* effectiveness and relevance for the modelling of EA’s, its effectiveness and relevance in modelling *low-level* (detailed) information is weak.

In fact, it is perhaps surprising that business users provided the support that they did in response to the question “LEAN is an effective language for modelling low-level (detailed) information.” This question was rated “4 → Disagree” by EA’s, but “3.4 → Agree” by business users. As LEAN is designed for high-level, conceptual modelling, it was expected that it would probably be seen as ineffective for low-level modelling. EA’s did take this view, but business users seemed less sure (in fact, four

respondents marked “Don’t know”). Nevertheless, the score of 3.4 was the highest rating attributed by business users to any question (indicating the lowest concurrence).

Business users concurred most highly with the assertions “LEAN leads users to think more deeply about the structures and relationships that exist” and “I would recommend LEAN for use by other enterprise architects”, which both scored a weighted average of 2.5 (exactly half way between Agree and Strongly Agree).

EA’s concurred most highly with the assertions “LEAN is an effective language for modelling high-level (conceptual) information”, “LEAN leads architects to think more deeply about the structures and relationships that exist” and “LEAN models convey meaning more precisely than the enterprise architecture models I previously used”, which all scored a weighted average of 2.5 (exactly half way between Agree and Strongly Agree). As mentioned previously, the highest score was in response to the question “Compared to other modelling languages I have used for enterprise modelling, LEAN is:” where the weighted average response for EA’s was “2 → Very easy to learn”.

7.1.6.2 Open Questions

Question 18 was unique to the EA’s survey: “What other languages have you used for enterprise modelling?” (Note, it is placed between questions 12 and 13 in the survey for continuity.)

Every EA had previously used UML for EA modelling supporting the view that UML is the most widely used, formal language, for EA modelling. No EA identified Archimate as an EA modelling language.

Architecture	13
Tool	11
Enterprise	10
Relationships	9
Change	8
Business	8
Help	7
Easy	7
Easily	7

Level	7
-------	---

Table 14 - Top Ten Words not in Wordlist

Text provided in answer to the open questions was analysed using Textalyser¹⁸ and TAPoR¹⁹. Question 1 (“What is your job title?”) and Question 2 (“What is your job function?”) was excluded from the analysis. Answers to all of the other open questions were included unless they consisted solely of the single word “no” or “yes”. Misspelt words have been corrected before analysis. Full stops were added to the end of each sentence if they were not already present and bullet points were removed.

General statistics are as follows:

- Total words: 1485
- Unique words: 558
- Average word frequency: 2.66

Table 14 shows the top ten words that are present in this text, excluding the words in the following wordlist: “a, an, of, the, in, on, at, and, or, for, to, it, not, I, is, with, be, that, as, but, have, was, use, lean, would, if, this, by, you, very”

Table 15 presents a concordance list for these high frequency words with a context length of five words.

<i>Architecture</i>
no other experience with enterprise architecture modelling tools making it not
relationships between components of the architecture .
higher level views of the architecture it proves very effective for
during the documentation of our architecture we may want some added
documentation maintenance of our IT architecture .
notations used today for Enterprise Architecture model drawings
simple mechanism for describing enterprise architecture without introducing the complexity inherent
to help CTOs plan enterprise architecture , explain the business goals of
RTA has an Enterprise Architecture Group
define RTA s current Enterprise Architecture .
An Enterprise Architecture team with no real power
<i>Tool</i>
of the task not the tool .

¹⁸ <http://textalyser.net/?q=iago.nac.net/~terbo/index.old.html>

¹⁹ <http://taporware.mcmaster.ca/~taporware/textTools/summarizer.shtml>

use simple high-level EA tool .
Not enough experience with the tool to have any suggestions to
An automated tool that enabled drill down or
Licensing of a LEAN modelling tool .
Lean like any other effective tool) will be effective in the
be improved like any other tool .
is supported by a drawing tool)
With an automated tool for LEAN this should be
should not become another modelling tool .
Enterprise
have no other experience with enterprise architecture modelling tools making it
disparate notations used today for Enterprise Architecture model drawings
a simple mechanism for describing enterprise architecture without introducing the complexity
potential to help CTOs plan enterprise architecture explain the business goals
modelling education services to CTO Enterprise Architects Management Executives
RTA has an Enterprise Architecture Group
to define RTA s current Enterprise Architecture
An Enterprise Architecture team with no real
analysis should bring about change Enterprise architects based on subjective and
grouping notation so that the enterprise can model for example what
Relationships
of capturing and conveying the relationships between components of the architecture
was excellent for depicting the relationships in the information architecture domains
that related more to the relationships in the infrastructure layers
each node easier to understand relationships .
Question Are the relationships connections directed only one way
business functions processes technology and relationships in a single easily readable
Defines relationships which can be easily enforced
I wish relationships were all objective i
Change
LEAN is to help with CHANGE .
5 years you need to change this and that
that you also need to change those other two linked systems
Bottom line for change is could lean associate cost
benefit analysis should bring about change .
It should assist change in EA in a real
is out of a proposed change) in one diagram
be very helpful to facilitate change in an environment where complex
Business
not previously exposed groups management business users etc
It effectively ties business functions processes and systems together
plan enterprise architecture explain the business goals of technology initiatives to
Effectively articulates business functions processes technology and relationships
that help to run a business .
UML BPEL Business Process Engineering Language OWL Ontology
how infrastructure and personnel support business processes
in an environment where complex business processes are supported by a

<i>Help</i>
Hoping the this will help gain the buy in from
It offers great potential to help CTOs plan enterprise architecture explain
Using LEAN to help tracking project status by management
LEAN is to help with CHANGE
LEAN should help architects identify areas that are
risks and real things that help to run a business
This would help at the inception level to
<i>Easy</i>
The diagrams were easy to read but I can
it is an efficient and easy to understand way of capturing
to read not overly complicated Easy to read not overly complicated
simple high-level EA tool Easy of use simple high-level
Extremely easy to learn
for LEAN this should be easy) I know relationship dependency is
<i>Easily</i>
that it can be relatively easily expanded and extended should resourcing
legends with the ability to easily jump between different areas of
that I know would be easily understood by CTOs CIOs
systems together into a neat easily understandable language
and relationships in a single easily
Defines relationships which can be easily enforced if it is supported
<i>Level</i>
of elaborating on the higher level views of the architecture it
the value as a high-level modelling tools looks very promising
Easy of use simple high-level EA tool
It enables high-level architectures to be visualised using
which operate at a lower level of abstraction
Useful at the highest level to show how infrastructure and
would help at the inception level to measure the scope of

Table 15 - Concordance List for High Frequency Words

7.1.6.3 Analysis of Models

Table 16 shows the frequency with which each type of node occurs within the ITD LEAN models.

	Node Type			
	Action	Agent	Resource	Rule
Number of Occurrences	75	40	147	33
Relative Frequency	25%	14%	50%	11%

Table 16 - LEAN Node Frequency Distribution

As Table 16 shows, the Rule node is used least frequently (11%). However, this node was found to be very useful for describing links between the system models and high-level organisational objectives. For instance, Rules were useful for describing organisational standards, requirements and objectives.

The fact that even the least used concept (Rule) was used a significant portion of the time suggests that all four LEAN concepts are valuable for this type of modelling. The fact that the concept of Resource was used so frequently (twice as much as its nearest rival, Action) suggests that the decomposition of this concept into more granular concepts may be justified. On the other hand, the support that LEAN provides for the development of hierarchies may negate the need to do this, highlighting the importance, perhaps, to ensure that enough attention is giving to ensure that appropriate Resource hierarchies are developed early in the modelling project.

Table 17 shows the frequency with which each type of *relationship* occurs within the ITD models. The tick marks show the legal relationships, while the number next to the tick shows the number of times that relationship was represented in the ITD LEAN models.

As Table 17 shows, some permissible relationships were not used at all in the ITD project. In particular, none of the “is a type of” relationships were used. These were designed to allow type-subtype relationships hierarchical relationships to be formed within each concept type.

Another type of hierarchical relationship, “reports to” was only used twice. Yet, the third, of only three types of hierarchical relationship that were supported, the “is a part of” relationship, was the most popular relationship of all. This relationship was used 78 times, primarily to create resource hierarchies (58 times).

This seems to indicate that EA modellers think strongly in terms component-subcomponent hierarchies when developing high-level models. These results *may* justify the exclusion of the “is a type of” and “reports to” relationships from the relationship set. Similarly, the “precedes” relationship, which was under-represented at only 3 occurrences, may warrant exclusion from the relationship set. This might help to simplify the relationship set further, making it easier to use while having little impact on the ability to create effective LEAN models. Instead, these relationships could be represented in domain specific models that are well equipped to represent these types of relationships. For example, application-modelling languages such as UML can well represent type-subtype relationships, process-modelling languages can represent temporal relationships and organisational charts can represent reporting relationships.

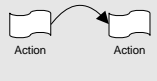

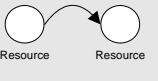
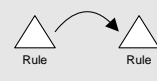
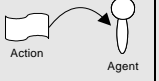
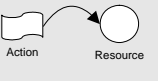
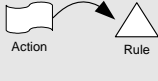


LEAN NODE PAIRINGS									
RELATIONSHIP SET	HOMOGENOUS PAIRINGS				HETEROGENOUS PAIRINGS				
									
<i>is a type of</i>	✓0	✓0	✓0	✓0	✗	✗	✗	✗	✗
<i>supports</i>	✗	✗	✓29	✗	✗	✗	✗	✗	✗
<i>interfaces with</i>	✗	✗	✓18	✗	✗	✗	✗	✗	✗
<i>is a part of</i>	✓23	✓5	✓50	✓0	✗	✗	✗	✗	✗
<i>precedes</i>	✓3	✗	✗	✗	✗	✗	✗	✗	✗
<i>reports to</i>	✗	✓2	✗	✗	✗	✗	✗	✗	✗
<i>performed by</i>	✗	✗	✗	✗	✓31	✗	✗	✗	✗
<i>uses</i>	✗	✗	✗	✗	✗	✓62	✗	✗	✗
<i>produces</i>	✗	✗	✗	✗	✗	✓18	✗	✗	✗
<i>complies with</i>	✗	✗	✗	✗	✗	✗	✓8	✓0	✓15
<i>applies to</i>	✗	✗	✗	✗	✗	✗	✓6	✓0	✓0
<i>supports goal</i>	✗	✗	✗	✗	✗	✗	✓6	✗	✗

Table 17 – Relationship Frequencies in ITD EA Models

7.1.7 Summary

The project that was carried out at ITD fits the criteria needed to assess the research hypothesis very well. The project covered a wide range of domains including application architecture, infrastructure architecture, business architecture and business strategy modelling. In addition, the project required modelling at various levels of abstraction, but with a focus on high-level modelling.

It should be remembered that, as reported in section 2.7, the prime indicator of the success of an EA modelling language appears to rest on how easy it is to use and understand. The premise of this research has been the observation that any successful integrated EA language will have to be easy to use. Attempts to develop a perfect unified language, at the expense of being human-centred have failed. It is better to forgo some semantic strength and ontological rigour in order to make the language more useable and intuitive. “A system architect can create a perfect EA model but it doesn’t matter if project teams can’t or won’t take advantage of it. The EA must be good enough, but does not need to be perfect.” (Kaisler et al., 2005)

The evidence gathered in this study supports the notion that LEAN is indeed easy to use as a tool for developing EA models. Survey responses such as the following support this finding:

- Extremely easy to learn
- Simple to use
- Simple to understand ...
- ... very effective and time-efficient
- I was very impressed with the number of systems that were able to be depicted in such a short amount of time.
- LEAN provides a simple mechanism for describing enterprise architecture without introducing the complexity inherent in languages which operate at a lower level of abstraction.

As reported by ITD in the final project report, it was asserted that the project delivered the following results:

The models have aided understanding of the current environment.

The models have generated discussion about various aspects of the current environment, leading to new insights.

The models have engendered ideas for simplification or augmentation of the current environment.

It is expected that these benefits will continue to expand as the models are distributed throughout the ITD and user community.

Appendix B – Final Project Report for UTS EA Project

The survey results appear to support these assertions.

In explaining the reason for developing LEAN models to other managers at the kick-off meeting for the project, Peter Demou (Manager Plans and Programs) summed up his understanding of LEAN by saying “It’s a kind of ‘cheat-sheet’ for managers”. The metaphor of a cheat-sheet conveys what the survey results have highlighted as the most significant aspects of LEAN: it is simple and effective.

The results provide a convincing argument for the efficacy of LEAN and support the research hypothesis.

7.2 Study Two: Re-Modelling a Public Domain Architecture

7.2.1 Introduction

In addition to the action research that was carried out and described in the previous chapter, the remodelling, using LEAN, of a large scale, existing EA, was identified as a potentially useful research method. This would allow a comparative analysis to be performed between the original EA models and the new LEAN models.

The number of available, real-world EA models, is limited. As previously mentioned, the development of enterprise architectures is generally viewed by organisations as providing strategic competitive advantage. The EA embodies the organisation’s unique strategies, goals and plans that it believes will give it an edge in the marketplace. As a result, these EA’s are generally seen as highly confidential, proprietary information,

making it very difficult to obtain this type of information for academic case study purposes. Another difficulty is that, as EA is still a relatively new technology, and despite its widespread adoption, few organisations have actually developed a cohesive and complete EA.

Fortunately, (for this research), the United States of America has enacted the Clinger-Cohen Act, which requires the development of enterprise architecture (EA) in Federal agencies. Section 5125 requires that agencies develop "a sound and integrated information technology architecture." Many of these EA's are public domain and are published on the Internet making them ideal source material for this research.

Interestingly, despite the fact that these publications often make references to EA 'models', many of the US published EA's contain few graphical models. Instead, these architectures tend to focus on the non-model aspects of EA such as principles, standards, policies and process. It can be argued that the presentation of these models as purely textual constructs reduces the ability of these architectures to persuade and convey information. Descriptions of the baseline and to-be architectures, although acknowledged as required parts of the EA development process, are mostly absent. Many of these EA's are works in progress with the process and methods described, but the architectural models missing. One could speculate that, without a human-centred, unified language for the development of EA level models, the completion of these enterprise architectures becomes a huge hurdle.

7.2.2 The USDoS Enterprise Architecture

The United States Department of State (USDoS) Information Technology Architecture (ITA) documents the high-level architecture of the USDoS and provides a good case for this research. It is a 'real world' EA, addressed to meet the demands of a large and complex organisation. The document that describes this EA is medium sized (approximately 40 pages) ensuring that the example is not trivial. In addition, the description that the USDoS provides for this document is congruent with typical EA definitions, making it suitable as an EA case study. This is illustrated by the following quote:

“The Department’s ITA provides guiding principles and standards to be applied when designing and implementing information services and specific systems for Department users. The ITA also specifies the major components of the technology infrastructure that need to be built to support the business requirements.” (United States Department of State, 1999, Section ES-1)

The case can be summarised as follows. The USDoS's current information systems are seen as fragmented, inconsistent and duplicative “islands of automation”. In response to this, it has developed an ITA that is expected to promote greater reusability, integration, portability and interoperability.


“The Department of State is modernizing its Information Technology (IT) infrastructure to support changing business needs and to take advantage of new technologies. In pursuit of this goal, the IRM Bureau has developed a high-level Information Technology Architecture (ITA) to guide the acquisition, development, and implementation of information technology ...” (United States Department of State, 1999)

The approach taken in this study was to analyse the USDoS ITA models and redevelop the majority of them as LEAN models. Models were not translated in the cases where their translation into LEAN would have been repetitive and add no research value.

It is recognised that in some cases it may not be beneficial, nor appropriate, to translate these models into LEAN graphs. Some of these models may have more explanatory and persuasive power in their original forms. However, their conversion to LEAN is designed to show that LEAN can be used for a wide range of high-level enterprise modelling tasks. There is, however, the potential advantage that, even if some individual models do lose some of their semantic or persuasive power, once *all* of the models are converted to LEAN, then it becomes possible to perform more detailed analysis than is possible when all the models are in different forms.

As well as converting all of the graphical models in the case study to LEAN models, many of the textual descriptions of the architecture and organisation have been modelled using LEAN. This is done to demonstrate the flexibility of LEAN. This approach also shows how a typical EA with a limited number of models could be translated into a comprehensive set of LEAN models with enough coded information to start to perform high-level design and re-engineering tasks for the organisation.

A copy of the USDoS ITA is provided in Appendix D – USDoS ITA. Parts of the document have been deleted to remove duplication (e.g. the executive summary).

To indicate which models and sections of text have been translated into LEAN, reference marks have been added - they look like this:  and contain a reference number that matches the corresponding LEAN model. The LEAN models are shown in Appendix E – USDOS ITA LEAN Models. All of the LEAN models created for this case were created using the Generic Relationship Set. No additional relationship types were created (or found necessary).

7.2.3 Analysis

A considerable amount of the information portrayed in the LEAN models is, by necessity, *inferred* from the narrative provided in the ITA document. Of course, in a real modelling situation there would be opportunity for feedback and clarification. However, for illustrative purposes this LEAN evaluation approach would appear to serve its purpose well enough.

Some of the LEAN models that have been produced contain ambiguous or redundant information. This is because the original USDoS ITA models are ambiguous or redundant (as an examination of the original models will quickly demonstrate). One of the reasons for this is that each of the models in the USDoS ITA appears to use a different schema, and none of the schemas are defined! In fact, close analysis of these models reveals that they contain many ambiguities and inconsistencies (in terms of the information presented as well as the schema used).

These inconsistencies seem to be easier to overlook in the original models. Perhaps the lack of formal definition reduces the ability of the reader to perform critical analysis, leading the viewer to identify the general concept without being bothered by the detail. On the one hand, this may be a highly beneficial aspect of this type of informal modelling. On the other hand, it does make any formal analysis and computer aided manipulation difficult until these vagaries are resolved. The lack of objective content in some of the USDoS ITA models would make it difficult to determine whether the models are correct or not as they are impossible to test in order to confirm or falsify

the content. An example of this is Model 2, which is supposed to convey the metaphors of systems as “islands of automation” and “stovepiped”.

In other cases, the use of graphics within the USDoS ITA merely distracts the reader from the fact that the model is actually semantically equivalent to little more than a textual list. Graphics, colours and ‘pleasing’ composition can give the impression that the model is rich in information, while in fact being quite devoid of any information other than the textual labels provided on the diagram. In these cases, conversion to LEAN appears to make this semantic paucity more apparent.

Table 18 shows the frequency with which each type of *relationship* occurs within the USDoS ITA models. The tick marks show the legal relationships, while the number next to the tick shows the number of times that relationship was represented in the USDoS ITA models.

In total, 180 relationships were represented, versus the 276 relationships that were represented in the ITD models. Yet, there are significantly more Resource-to-Resource relationships in the USDoS ITA models. This indicates that the USDoS ITA models are weighted heavily towards the representation of structural relationships, particularly hierarchical relationships, between enterprise resources.

In comparison to the original USDoS ITA models, the LEAN models may seem to be relatively complicated vehicles for describing architectural information. There are two points to be made about this. Firstly, while it *may* seem more cumbersome to model the information using LEAN versus text or other modelling approaches (even informal ones), LEAN carries with it the benefits of being rigorously defined. This means that once the LEAN model is created, it can then be processed in many ways to generate additional information. In this respect, the LEAN models are richer than the original information on which they are based.

Secondly, if we look at other approaches for *formally* modelling enterprise information (as opposed to the informal, non-defined models used in the US DoS ITA), we can see that LEAN is not so cumbersome at all. Take for example the following top-level generic activity model, modelled using IDEF0 (Presley et al., 2001) shown in Figure 25.

Clearly, this model is far more detailed and intricate than most of the LEAN models provided in this Study, and yet it is similarly intended for use by EA stakeholders.

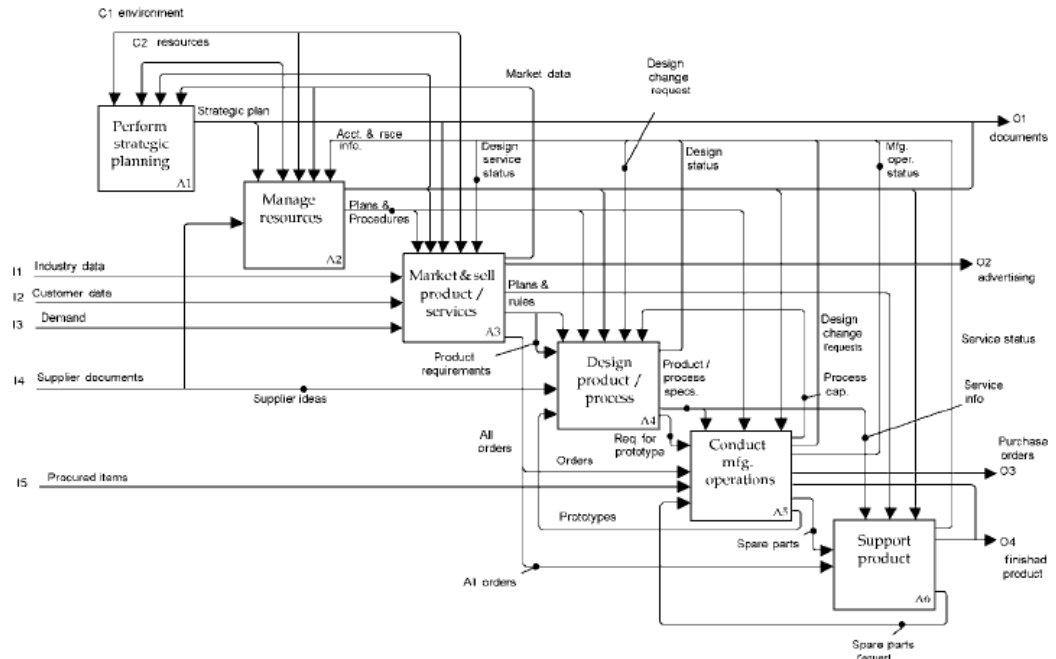


Figure 25 - Top Level Generic Activity Model

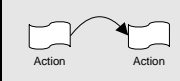

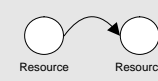
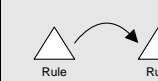

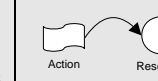



LEAN NODE PAIRINGS									
RELATIONSHIP SET	HOMOGENOUS PAIRINGS				HETEROGENOUS PAIRINGS				
									
<i>is a type of</i>	✓0	✓2	✓7	✓0	x	x	x	x	x
<i>supports</i>	x	x	✓40	x	x	x	x	x	x
<i>interfaces with</i>	x	x	✓5	x	x	x	x	x	x
<i>is a part of</i>	✓12	✓1	✓65	✓0	x	x	x	x	x
<i>precedes</i>	✓8	x	x	x	x	x	x	x	x
<i>reports to</i>	x	✓0	x	x	x	x	x	x	x
<i>performed by</i>	x	x	x	x	✓8	x	x	x	x
<i>uses</i>	x	x	x	x	x	✓10	x	x	x
<i>produces</i>	x	x	x	x	x	✓17	x	x	x
<i>complies with</i>	x	x	x	x	x	x	✓3	✓0	✓9
<i>applies to</i>	x	x	x	x	x	x	✓2	✓0	✓0
<i>supports goal</i>	x	x	x	x	x	x	✓0	x	x

Table 18 - Relationship Frequencies in USDoS ITA

7.2.4 Summary

In many cases, it is quite surprising just how much information, represented as LEAN models, can be gleaned from what seems to be a simple textual description. This suggests that LEAN is able to capture, and formalise, quite subtle grammatical semantics.

It should be noted that the goal of LEAN is *not* to make redundant more informal methods of describing enterprise level architectural concepts, but to bridge the gap between informal and formal representations. Informal models are often more naturally generated from the information at hand, and in some cases may have better communicative powers. Their weakness is their lack of semantic integrity and the inability to transform these models logically into more detailed designs. The goal of LEAN is to augment these informal approaches with a more rigorous standard that supports clearer communication at the highest conceptual levels, while also leading logically into more specialised domain architectures.

7.3 Evaluation

This research was designed to investigate the following hypothesis: *It is possible to develop a human-centred modelling language for creating unified models that span heterogeneous domains of an enterprise architecture.*

The development of a single modelling language that meets the criteria set out in the hypothesis will prove the hypothesis true. LEAN was developed as a modelling language that might meet these criteria. LEAN was developed by first creating theoretical principles for developing unified enterprise metaphors, developing a methodology to use these metaphors to construct unified modelling languages, and then applying the methodology to produce one instance of a unified EA modelling language (LEAN).

The approach taken to test LEAN was to break the hypothesis down into two separate research questions:

1. Can we develop a modelling language for creating unified models that span heterogeneous domains of an EA?

2. Can such a language be designed to be human-centred?

If these two research questions are answered in the affirmative, then we have also provided evidence that the theory and methodology developed in this thesis have efficacy for developing a *range* of unified EA modelling languages.

With respect to the first research question, two disparate studies were conducted to determine if the LEAN modelling language could be used to create unified models that span heterogeneous domains of an EA. Study One involved the development of EA models by managers and IT specialists within an operating enterprise that had a large scale, complex IT environment. Study Two involved the reformulation of EA models that were originally produced by the US Department of States and placed in the public domain. The results from Study One show that the LEAN language *can* be used to create unified models that span heterogeneous domains of an EA. One of the stated objectives of the ITD EA project was to show the interrelationships between the different domain architectures.

The domains that were subsequently modelled in this project included Infrastructure, Applications and Business architecture domains. The survey that was issued to the project stakeholders and independent analysts following project completion confirmed that LEAN was used *successfully* to capture information across all of the technical and business domains of interest. Furthermore, answers to the open survey questions include numerous comments on the effectiveness of LEAN to show the relationships between disparate components of the EA.

The results from Study Two provide a number of insights. Firstly, it was observed that the original USDoS ITA models were lacking in semantic integrity (a wide variety of schema was used, none of which were defined and there were many inconsistencies within, and between the models) and expressiveness (the models conveyed relatively little meaning and were used predominantly to illustrate structural relationships between Resources, and there were many ambiguities in the models).

Conversion of the USDoS ITA models into LEAN models showed that the structures available in LEAN were sufficient to represent the structures presented in the original models, despite the wide variety of notations used in those models. LEAN could also be used to represent information structures represented as text in the original USDoS ITA, thus making this information visual.

With respect to the second research question, Study One showed that LEAN *is* human-centred. The survey that was issued to the project stakeholders and independent analysts following project completion confirmed that LEAN was easy to use and very easy for enterprise architects to learn. The answers to the open survey questions include numerous comments on the simplicity of LEAN. The open survey questions also included a number of comments that indicated that the graphic nature of the LEAN models supported effective and easy visualisation of the EA and this was supported by the positive answer to the closed question “LEAN leads users to think more deeply about the structures and relationships that exist”.

Together, Study One and Study Two provide solid evidence that the two research questions can both be answered in the affirmative. Therefore, we can conclude that *it is possible to develop a human-centred modelling language for creating unified models that span heterogeneous domains of an enterprise architecture*. The research presented in this thesis demonstrates that the stated hypothesis can be answered in the affirmative.

8 CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

"The ideal architect should be a man [or woman] of letters, a mathematician, familiar with historical studies, a diligent student of philosophy, acquainted with music, not ignorant of medicine, learned in the responses of jurisconsults, familiar with astronomy and astronomical calculations." Vitruvius, circa 25 BC

In this chapter, conclusions from this programme of work are provided, along with the implications of this research. A number of opportunities for further research have been identified and are described here.

8.1 Conclusions

As a scientific discipline, Enterprise Architecture (EA) is still in its infancy. However, the development of an EA is widely recognised as an important endeavour that can provide an enterprise with significant strategic competitive advantage. As IT environments become more complex and heterogeneous, the importance of EA's will increase.

There are several modelling techniques that are specially designed for describing EA's, and depending on the particular EA frameworks, standards and methodologies being used, one or more of these modelling techniques may be recommended. However, all of the formal contemporary approaches to EA modelling suffer from one or more of the following drawbacks: either they are too complex for real-world application, or they can only describe limited domains or specific industries. As a result, Enterprise Architects have to learn, and use, a wide variety of modelling languages in order to cover the full EA domain encompassing application, infrastructure, information and business architectures. Alternatively, the Enterprise Architect invents an undefined notation that subsequently lacks semantic integrity. Without a human-centred, unified EA language, there is no way to develop coherent and consistent EA's that support the development of a shared understanding of the enterprise's goals and objectives (Bernus, 2001).

A novel approach has been used in this thesis to develop a unified EA modelling language that does not suffer from the complexity that haunts other unified EA languages. Through an understanding of metaphor type hierarchies, it was shown that

metaphors can be developed that describe a wide variety of enterprises at various levels of abstraction. This was referred to as a 'unified metaphor'. A methodology was then presented for developing this metaphor into an ontology, and subsequently formalising and codifying this ontology to produce a high-level, unified, EA modelling language.

In order to evaluate this theory, the methodology was applied to develop a unified modelling language based on the metaphor 'an enterprise is a society'. Giddens theory of structuration was used as a basis for the development of a society ontology, and this was further developed into graphical modelling language. Since visual systems offer distinct advantages over non-visual systems for the development of effective cognitive maps, the development of a *graphical* language was seen as being of key importance. The language that was developed is referred to as the Lightweight Enterprise Architecture Notation (LEAN).

Validation of LEAN as a unified and human-centred language then took place. In order to provide strong results, a mixed-methods research approach was used that combined experimental research with action research. Three separate studies were then performed: two formal studies, and one informal study.

This research was designed to test the hypothesis that it was possible to develop a human-centred EA modelling language that was unified. Other unified EA modelling languages already exist, but they are not human-centred, and consequently, they have not been used for EA modelling outside of academic circles. To test this hypothesis it was necessary to first confirm that LEAN was indeed unified, and then to test whether it was also human-centred. In combination, these attributes would demonstrate that the theory and methodology developed in this thesis was successful in developing a useful unified EA modelling language. It could then be inferred, although it remains unproven at this stage, that this theory and methodology could also be used to develop a *range* of unified EA modelling languages.

The formal studies produced results that support the research hypothesis. Study One involved running an EA project in the IT department of a large, diverse enterprise. This study provided strong evidence that LEAN is a unified EA modelling language that can

be used to model a wide variety of structures at various levels of abstraction, and that it is human-centred. Business users and Enterprise Architects were surveyed on a wide range of attributes in order to determine the efficacy of LEAN for EA modelling and they responded in the *positive for every tested attribute*. In addition, comments provided in response to open questions confirmed that users found LEAN easy to learn and simple to use: essential attributes of effective unified EA modelling languages.

Study Two, based on a complex real-world public domain EA, added weight to the premise that LEAN is unified and can be used to model a wide variety of constructs that would typically be represented in an enterprise architecture.

The theory and methodology developed in this thesis is a highly original approach to the development of a unified EA modelling language. Traditional approaches to this problem take a 'bottom up' perspective that leads to highly complex solutions.

The approach taken here is quite novel in that it combines elements of cognition, linguistics, social theory and technology to produce highly flexible EA languages that are both integrated and human-centred. This new theory and methodology provides a substantive contribution to the field of computer science. Additionally, the acknowledgement by EA stakeholders and experts that LEAN is a human-centred, unified EA modelling language provides industry with a valuable new modelling language for developing EA's and achieving competitive advantage.

8.2 Future Research Directions

Since EA is still an immature discipline, it is not surprising that the findings presented in this thesis provide fertile ground for further research. In terms of the theoretical foundations developed in this research, there are a number of areas that deserve further investigation.

The use of the theoretical approach and methodology presented in this thesis to develop languages based on *other* unifying metaphors would provide valuable insights. It would strengthen the proposition that this methodology can be used to develop an unlimited set of unified modelling languages. Perhaps more importantly however, the development

of additional languages would allow a comparative analysis of these languages to be performed. This would provide data on the characteristics that make such unified languages effective and would allow the development of *optimised* unified EA modelling languages.

Study Three demonstrated how an effective unified EA modelling language based on an enterprise metaphor may have properties that allow it to be used to structure systems at various levels of abstraction. This would provide greater *congruence* between models at various layers of abstraction. Further research investigating these properties of metaphor based unified modelling languages is warranted. Greater alignment between high-level business objectives and the information systems that support these objectives will lead to better systems integration (leading to easier maintenance and better useability), and the realisation of greater business value as decisions made at strategic levels will be implemented more consistently and completely.

In relation to the LEAN modelling languages, there are several areas that can be researched and developed further. Firstly, it would be valuable to extend the research presented in this thesis by using LEAN in additional EA modelling projects across a variety of industries. This would provide further information on the *generality* of LEAN.

The LEAN relationship set also warrants refinement. This is likely to be an ongoing process as identification of the most useful generic relationships is identified through a heuristic process.

The development of mechanisms and procedures for integrating LEAN with other enterprise ontologies such as TOVE and the IDEF ontologies could be developed. This would allow the reuse of knowledge from these ontologies and may improve the utility of LEAN.

An investigation of other, non-EA, cross-disciplinary domains could be valuable in order to determine the characteristics of these domains and the potential for using similar approaches for modelling these domains. This may greatly extend the value of the approach developed in this thesis to other problems.

The LEAN modelling tool could also be further developed in the following areas:

- Simulation features and metric analysis tools (e.g. time, support and cost analysis) could be added to leverage the potential of the stored information for the purposes of strategic planning.
- The tool could be expanded as an EA repository by supporting the storage, organisation and retrieval of other types of documents and images.

Interfaces, such as an XML interface, could be built to allow information exchange with other tools and systems.

8.3 Closing Remarks

“... in the 21st century it [Enterprise Architecture] will be the determining factor, the factor that separates the winners from the losers, the successful and the failures, the acquiring from the acquired, the survivors from the others.” (Zachman, 1997)

While specialised languages have been developed for modelling specific domains such as application, infrastructure and network architectures, there are no human-centred languages that support the creation of high-level, conceptual systems models that extend across all IT domains. Without such a language, there is no way to develop coherent and consistent EA models. While there have been several attempts to solve this problem, none can be said to be highly successful. Indeed, some researchers have described the problem as intractable, while others declare its solution, an open problem (Dewhurst et al., 2002) (Zelm and Kosanke, 2001) (Lankhorst, 2005 p. 56).

In fact, the problem *can* be solved as long as one understands the criteria upon which the success of such a language rests. The key to developing a useful EA modelling language is summed up by Solberg as follows: "Enterprise models are useful only if they are used. They will be accepted by users as a tool if they are simple to understand, easy to use, computer-supported, and if they provide a realistic image of the reality. This explains the failure of many approaches proposed in the past ..." (2000, p.184)

As the profile of EA increases, those in charge of the direct lines of business will seek hands-on ownership of the EA. EA's are too important to leave to the IT specialists! Just as businesses sought to take greater control over application design and

development in the 1980's, we will see EA ownership devolve to those who are making operational and strategic decisions upon which the business will succeed or fail.

This devolution of ownership will drive the way we design and develop EA's. They will need to be designed with a business perspective, rather than a blinkered focus on technology. The focus of EA's will expand from the management and analysis of computer systems, to the management and analysis of *information* systems that support the identification of tactical and strategic opportunities. The scope will extend from networks, servers and data structures, to incorporate concepts such as time to market, cost effectiveness, change management, and so on. In addition, the EA systems will need to be designed appropriately so that business planners and developers can leverage the models to achieve corporate goals.

Ultimately, the goal of an EA is to reduce complexity and increase agility (Fayad et al., 2002). It is only by understanding this concept that the development of *effective* EA modelling languages can be realised. "... simplicity enhances an organization's ability to use technology more effectively." (Theuerkorn, 2005 p.138)

As new technologies continue to disrupt business models and challenge the ability of IT departments to manage and exploit opportunities for change, the role of EA will continue to become more important (Khoury, 2006b). This makes the development of an effective integrated EA modelling language imperative. LEAN provides such a language, while the theory and methodology presented in this thesis provides the potential to develop a *range* of languages that meet these needs.

9 BIBLIOGRAPHY

- ALLEN, R. B. (1997) Mental Models and User Models. IN HELANDER, M., LANDAUER, T. K. & PRABHU, P. (Eds.) *Handbook of Human-Computer Interaction*. 2nd ed., Elsevier Science.
- AMBLER, S. W. (2004) Architecture and Architecture Modeling Techniques: Bringing data professionals and application developers together, www.agiledata.org, <http://www.agiledata.org/essays/enterpriseArchitectureTechniques.html>.
- ANTONIOU, G. & HARMELEN, F. V. (2004) *A Semantic Web Primer*, London, England, The MIT Press.
- ARLOW, J. & NEUSTADT, I. (2003) *Enterprise Patterns and MDA: Building Better Software with Archetype Patterns and UML*, Boston, MA, Addison Wesley Professional.
- ARMOUR, F. J., KAISLER, S. H., GETTER, J. & PIPPIN, D. (2003) A UML-driven enterprise architecture case study. *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*. Hawaii.
- BAKER, D. C. & JANISZEWSKI, M. (2006) 7 Essential Elements of EA, http://www.diamondconsultants.com/PublicSite/ideas/perspectives/downloads/INSIGHT%20-%20Seven%20Essential%20Elements%20of%20Enterprise%20Architecture_single.pdf.
- BASKERVILLE, R. L. (1999) Investigating Information Systems with Action Research. *Communications of the Association for Information Systems*, 2/19.
- BERNARD, S. A. (2004) *An Introduction to Enterprise Architecture*, Bloomington, Indiana, AuthorHouse.
- BERNUS, P. (2001) Some thoughts on enterprise modelling. *Production Planning and Control*, 12, 110-118.
- BERNUS, P. (2003) Enterprise models for enterprise architecture and ISO9000:2000. *Annual Reviews in Control*, 27, 211-220.
- BERNUS, P. & NEMES, L. (1994) A Framework to Define a Generic Enterprise Reference Architecture and Methodology. *Minutes, Eighth Workshop Meeting, IFAC/IFIP Task Force on Architectures for Enterprise Integration*. Vienna, Austria.
- BEZNOSOV, K. (1998) Architecture of Information Enterprises: Problems and Perspectives. *Architecture of Information Enterprises: Problems and Perspectives. Written for the "Advanced Topics in Software Engineering" seminar given by Dr. Michael Evangelist during spring of 1998 at School of Computer Science, Florida International University*.
- BIEMANS, F. P. M., LANKHORST, M. M., TEEUW, W. B. & WETERING, R. G. V. D. (2001) Dealing with the Complexity of Business Systems Architecting. *Systems Engineering*, 4, 118-133.
- BLACK, M. (1962) *Models and Metaphors: Studies in Language and Philosophy*, London, Cornell University Press.
- BLACK, M. (1979) More about Metaphor. IN ORTONY, A. (Ed.) *Metaphor and Thought*. London, Cornell University Press.
- BLACKWELL, A. F. (1998) Metaphor in Diagrams. *Darwin College*. Cambridge, University of Cambridge.
- BOAR, B. H. (1999) *Constructing Blueprints for Enterprise IT Architectures*, New York, Wiley Computer Publishing.

- BRYANT, C. G. A. & JARY, D. (Eds.) (1991) *Giddens' theory of structuration : a critical appreciation*, London; New York, Routledge.
- CHALMETA, R., CAMPOS, C. & GRANGEL, R. (2001) References architectures for enterprise integration. *Journal of Systems and Software*, 57, 175-191.
- CHECKLAND, P. (1981) *Systems Thinking, Systems Practice*, London, John Wiley & Sons.
- CHECKLAND, P. & SCOLES, J. (1990) *Soft Systems Methodology in Action*, New York, Wiley.
- CHEN, C. (1999) *Information Visualisation and Virtual Environments*, London, Springer-Verlag.
- CHEN, D. & VERNADAT, F. (2004) Standards on enterprise integration and engineering - state of the art. *International Journal of Computer Integrated Manufacturing*, 17, 235-253.
- CLARKE, J., MODGIL, C. & MODGIL, J. (Eds.) (1990) *Anthony Giddens: Consensus and Controversy*, Brighton, UK, Falmer Press.
- CLEMENTS, P. C. (1996) A Survey of Architecture Description Languages. *Eighth International Workshop on Software Specification and Design*. Germany.
- COOK, S. (2005) The Architecture of UML, Object Management Group (OMG), http://www.omg.org/news/meetings/workshops/presentations/uml_presentations/5-1%20Cook%20-%20SJC.pdf.
- CORNELISSEN, J. P. (2005) Beyond Compare: Metaphor in Organization Theory. *Academy of Management Review*, 30, 751-764.
- COSTIKYAN, G. (1994) I Have No Words & I Must Design. *Interactive Fantasy* #2.
- COUPRIE, D., GOODBRAND, A., LI, B. & ZHU, D. (2004) Soft Systems Methodology, <http://sern.ucalgary.ca/courses/seng/613/F97/grp4/ssmfinal.html>.
- CYRE, W. R. (1997) Capture, Integration, and Analysis of Digital System Requirements with Conceptual Graphs. *IEEE Transactions on Knowledge and Data Engineering*, 9.
- DENFORD, M., SOLOMON, A., LEANEY, J. & O'NEILL, T. (2004) Architectural Abstraction as Transformation of Poset Labelled Graphs. *Journal of Universal Computer Science*, 10, 1408-1428.
- DEWHURST, F. W., BARBER, K. D. & PRITCHARD, M. C. (2002) In search of a general enterprise model. *Management Decision*, 5, 418-427.
- DIEBERGER, A. (1994) Navigation in Textual Virtual Environments using a City Metaphor. *Faculty of Technology and Sciences*. Vienna, Vienna University of Technology.
- DIEBERGER, A. & FRANK, A. U. (1998) A City Metaphor to Support Navigation in Complex Information Spaces. *Journal of Visual Languages and Computing*, 9, 597-622.
- DIJKSTRA, E. W. (1976) *A Discipline of Programming*, Englewood Cliffs, New Jersey, Prentice Hall.
- DIJKSTRA, E. W., DENNING, P. J., PARNAS, D. L., SCHERLIS, W. L., EMDEN, M. H. V., COHEN, J., HAMMING, R. W., KARP, R. M. & WINOGRAD, T. (1989) A debate on teaching computing science. *Communications of the ACM*, 32, 1397 (18).
- DOUMEINGTS, G. & DUCQ, Y. (2001) Enterprise modelling techniques to improve efficiency of enterprises. *Production Planning and Control*, 12, 146-163.

- DRUCKER, P. F. (1969) *The Age of Discontinuity: Guidelines to Our Changing Society*, New York, Harper & Row.
- DRUCKER, P. F. (1992) The New Society of Organizations. *Harvard Business Review*, September-October 1992, 95-104.
- EVERNDEN, R. (1996) The Information Framework. *IBM Systems Journal*, 35, 37-68.
- FAYAD, M., HAMU, D. & BRUGALI, D. (2002) Enterprise frameworks. *Software - Practice and Experience*, 32, 735-736.
- FOWLER, M. (2003) The New Methodology, <http://martinfowler.com/articles/newMethodology.html>.
- FOX, M. S., CHIONGLO, J. F. & FADEL, F. G. (1993) A Common-Sense Model of the Enterprise. *Second Industrial Engineering Research Conference*. Norcross, GA, USA, Institute for Industrial Engineers.
- FOX, M. S. & GRUNINGER, M. (1997) On Ontologies And Enterprise Modelling. *Proceedings of the International Conference on Enterprise Integration Modelling Technology*. Springer Verlag.
- FOX, M. S. & GRUNINGER, M. (1998) Enterprise Modeling. *AI Magazine*, 19, 3, 109 - 121.
- FRANKEL, D. S. (2003) *Model Driven Architecture: Applying MDA to Enterprise Computing*, Indianapolis, Wiley Publishing.
- FRASER, J. & TATE, A. (1995) The Enterprise Tools Set - An Open Enterprise Architecture. *Proceedings of Workshop on Intelligent Manufacturing Systems (IMS), International Joint Conference on Artificial Intelligence (IJCAI-95)*.
- FREDERICK P. BROOKS, J. (1995) *The Mythical Man-Month: Essays on Software Engineering*, Reading, Massachusetts, Addison-Wesley.
- GAINES, B. R. & SHAW, M. L. G. (1995) Collaboration through concept maps. *Proceedings of CSCL95: Computer Supported Cooperative Learning*, Bloomington.
- GARDINER, M. M. & CHRISTIE, B. (Eds.) (1987) *Applying Cognitive Psychology to User-Interface Design*, Chichester, John Wiley & Sons.
- GARTNER, A. L.-. (2002) The future of computer interfaces: Human-Computer Interfaces From 2003 to 2012, ZDNet, <http://techupdate.zdnet.com/techupdate/stories/main/0%2C14179%2C2901050%2C00.html>.
- GEUS, A. D. (1997) *The Living Company*, Boston, Harvard Business School Press.
- GHYCZY, T. V. (2003) The Fruitful Flaws of Strategy Metaphors. *Harvard Business Review*, 86-94.
- GIDDENS, A. (1984) *The Constitution of Society: Outline of the Theory of Structuration*, Cambridge, Polity Press.
- GOMEZ-PEREZ, A. (1995) Some Ideas and Examples to Evaluate Ontologies. *Proceedings of the Eleventh Conference on Artificial Intelligence Applications*. Los Alamitos, CA, IEEE Computer Society Press.
- GREFEN, P. W. P. J. (1997) Modeling architectures of complex information systems, School voor Informatie- en KennisSystemen, <http://www.siks.nl/ond/mod/grefen.html>.
- GRUBER, T. R. (1993) Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal Human-Computer Studies*, 907-928.
- GRUNINGER, M. & FOX, M. S. (1996) The Logic of Enterprise Modelling. IN BERNUS, P. & NEMES, L. (Eds.) *Modelling and Methodologies for Enterprise*

- Integration: Proceedings of the IFIP TC5 Working Conference on Models and Methodologies for Enterprise Integration*. London, Chapman & Hall.
- GUBA, E. G. & LINCOLN, Y. (1989) *Fourth Generation Evaluation*, Newbury Park, Sage Publications.
- GUSTAS, R. (2005) Inference Rules of Semantic Dependencies in the Enterprise Modelling. IN FUJITA, H. & MEJRI, M. (Eds.) *Proceedings of the fourth SoMeT_W05*. Tokyo, Japan, IOS Press.
- HALASZ, F. & MORAN, T. P. (1982) Analogy considered harmful. *Human Factors in Computer Systems Conference*. Gaithersburg, Maryland, ACM.
- HAMMOND, N. V. & ALLISON, L. J. (1987) The Travel Metaphor as Design Principle and Training Aid for Navigating Around Complex Systems. IN DIAPER, D. (Ed.) *Proceedings of the Third Conference of the British Computer Society*. University of Exeter, Cambridge University Press.
- HAUGHEY, T. (2005) Modeling Hierarchies, <http://www.tdan.com/special031.htm>.
- HENDERSON-SELLERS, B. & BULTHUIS, A. (1998) *Object-Oriented Metamethods*, New York, Springer-Verlag.
- HIGHSMITH, J. (2002) *Agile Software Development Ecosystems*, Boston, MA, Pearson Education, Inc.
- HILLIARD, R. (1999a) Aspects, Concerns, Subjects, Views, ... *Submission to the OOPSLA'99 Workshop on Multi-Dimensional Separation of Concerns in Object-Oriented Systems*.
- HILLIARD, R. (1999b) Views and Viewpoints in Software Systems Architecture. *Position paper for the First Working IFIP Conference on Software Architecture (WICSA 1)*. San Antonio, TX.
- HILLIARD, R. (2000) IEEE-Std-1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems, <http://www.enterprise-architecture.info/Images/Documents/IEEE%201471-2000.pdf>.
- HIRST, G. (2003) Ontology and the Lexicon. IN STAAB, S. & STUDER, R. (Eds.) *Handbook on Ontologies in Information Systems*. Berlin, Springer.
- HO, C.-S., HONG, Y.-C. & KUO, T.-S. (1986) A Society Model for Office Information Systems. *ACM Transactions on Office Information Systems*, 4, 104-131.
- IACOB, M.-E., JONKERS, H., LANKHORST, M., BUUREN, R. V., GROENEWEGEN, L., CHEUNG, K. H., BONSANGUE, M. & KAMPENHOUT, N. V. (2002) *State of the Art in Architecture Concepts and Description Vn 1.0*, Telematica Instituut (TI).
- IDEF (1992) *IDEF1 Information Modeling*, College Station, Texas, Knowledge Based Systems, Inc.
- IDEF (1993) *Integration Definition for Function Modeling (IDEF0) Draft*, Federal Information Processing Standards Publication 183.
- IEEE (2002) IEEE Recommended Practice for Architectural Description of Software-Intensive Systems (IEEE Std 1471), http://www.pithecantropus.com/~awg/public_html/.
- INDURKHYA, B. (1994) The Thesis That All Knowledge Is Metaphorical and Meanings of Metaphor. *Metaphor and Symbolic Activity*, 9, 61-63.
- INSTITUTE FOR ENTERPRISE ARCHITECTURE DEVELOPMENTS (2004) Trends in Enterprise Architecture 2004: How are Organizations Progressing?, <http://www.enterprise->

- architecture.info/Images/EA%20Survey/EA%20Survey%202004%20IFEAD.PDF.
- INSTITUUT, T. (2004) Archimate Language Primer. Enschede, Telematica Instituut.
- INSTITUUT, T. (2005a) Archimate, Telematica Instituut, <http://www.telin.nl/projecthome.cfm?id=48&language=en>.
- INSTITUUT, T. (2005b) Architecture Language Reference Manual, Telematica Instituut, https://doc.telin.nl/dscgi/ds.py/Get/File-31626/D2.2.2b_Architecture_Language_Reference_Manual_v4.0.pdf.
- ISO (1995) ITU-T Rec. X.901 | ISO/IEC 10746-1 ODP Reference Model Part 1. Overview.
- ITU (1995) ITU Recommendation X.903 | ISO/IEC 10746-3, Open Distributed Processing - Reference Model - Part 3: Architecture.
- ITU (1996) ITU Recommendation X.901 | ISO/IEC 10746-1: Open Distributed Processing - Reference Model - Part 1: Overview.
- ITU (1997) ITU Recommendation X.904 | ISO/IEC 10746-4, Open Distributed Processing - Reference Model - Part 4: Architectural Semantics.
- JACKSON, P. & WEBSTER, W. R. (2007) The Social Reality of Business Activity: A Contingent Methodology for Knowledge Elicitation and Mapping. *International Journal of Knowledge Management*, 3, 49-65.
- JOHNSON, G. J. (1994) Of metaphor and the difficulty of computer discourse. *Communications of the ACM*, 37, 97-102.
- JONKERS, H., BUUREN, R. V., ARBAB, F., BOER, F. D., BONLANGUE, M., BOSMA, H., DOEST, H. T., GROENEWEGEN, L., SCHOLTEN, J. G., HOPPENBROUWERS, S., IACOB, M.-E., JANSSEN, W., LANKHORST, M., LEEUWEN, D. V., PROPER, E., STAM, A., TORRE, L. V. D. & ZANTEN, G. V. V. (2003) Towards a Language for Coherent Enterprise Architecture Descriptions. *Seventh IEEE International Enterprise Distributed Object Computing Conference (EDOC'03)*. Brisbane, Queensland, Australia.
- KAISLER, S. H., ARMOUR, F. & VALIVULLAH, M. (2005) Enterprise Architecting: Critical Problems. *38th Hawaii International Conference on System Sciences*. Hawaii, IEEE.
- KHOURY, G. (2006a) The Art of Guerrilla IT. *Australian Technology and Business*, 52-56.
- KHOURY, G. (2006b) Uncovering the Matrix. *Australian Technology and Business*, February 2006.
- KHOURY, G. R., SIMOFF, S. & DEBENHAM, J. (2005) Modelling Enterprise Architectures: An Approach Based on Linking Metaphors and Ontologies. IN MEYER, T. & ORGUN, M. A. (Eds.) *Australasian Ontology Workshop (AOW 2005)*. Sydney, Australia, ACS.
- KHOURY, G. R. & SIMOFF, S. J. (2003) Elastic Metaphors: Expanding the Philosophy of Interface Design. *Conferences in Research and Practice in Information Technology: Computers and Philosophy 2003*, 37.
- KHOURY, G. R. & SIMOFF, S. J. (2004) Enterprise Architecture Modelling Using Elastic Metaphors. *Conferences in Research and Practice in Information Technology: Computers and Philosophy 2003*, 31.
- KHOURY, G. R. & SIMOFF, S. J. (2005) Philosophical Foundations for a Unified Enterprise Modelling Language. *Computers and Philosophy CAP2005*. Bangkok, Thailand.

- KNIGHT, C. (2002) System and Software Visualisation. IN CHANGE, S. K. (Ed.) *Handbook of Software Engineering and Knowledge Engineering*. World Scientific.
- KNOWLEDGE BASED SYSTEMS (2004a) IDEF5 Ontology Description Capture Method, Knowledge Based Systems, Inc, <http://www.idef.com/IDEF5.html>.
- KNOWLEDGE BASED SYSTEMS, I. (2004b) IDEF Integrated Definition Methods, <http://www.idef.com/>.
- KOLP, M., DO, T. T., FAULKENER, S. & HOANG, T. T. H. (2005) Architectural Styles and Patterns for Multi-Agent Systems. IN KHOSLA, R., ICHALKARANJE, N. & JAIN, L. C. (Eds.) *Design of Intelligent Multi-Agent Systems: Human-Centredness, Architectures, Learning and Adaption*. Berlin Heidelberg, Springer-Verlag.
- KREMER, R. & GAINES, B. R. (1996) Embedded Interactive Concept Maps in Web Documents. IN MAURER, H., ED. (Ed.) *Proceedings of WebNet96*. Charlottesville, VA, Association for the Advancement of Computing in Education.
- L'ABBATE, M. & HEMMJE, M. (1998) Virgillio - The metaphor definition tool. *GMD Report*, 15, pp.47.
- LAKOFF, G. (1993) The Contemporary Theory of Metaphor. IN ORTONY, A. (Ed.) *Metaphor and Thought*. 2nd ed. Cambridge, Cambridge University Press.
- LAKOFF, G. (2004) *Don't Think of an Elephant – Know Your Values and Frame the Debate*, Melbourne, Scribe Publications.
- LAKOFF, G. & JOHNSON, M. (1980) *Metaphors we live by*, Chicago, University of Chicago Press.
- LAKOFF, G. & NUNEZ, R. E. (2000) *Where Mathematics Comes From*, New York, Basic Books.
- LANKHORST, M. (2005) *Enterprise Architecture at Work: Modelling, Communication, and Analysis*, Berlin Heidelberg, Germany, Springer-Verlag.
- LUDEWIG, J. (2003) Models in software engineering - an introduction. *Software and Systems Modelling*, 2, 5-14.
- MAHER, M. L., SIMOFF, S., GU, N. & LAU, K. H. (2000) Designing Virtual Architecture. *Proceedings of CAADRIA 2000*. Singapore.
- MAIER, M. W. & RECHTIN, E. (2000) *The Art of Systems Architecting*, Boca Raton, CRC Press.
- MARAKAS, G. M., JOHNSON, R. D. & PALMER, J. W. (2000) A theoretical model of differential social attributions toward computing technology: when the metaphor becomes the model. *International Journal Human-Computer Studies*, 52, 719-750.
- MAYER, R. J., MENZEL, C. P., PAINTER, M. K., DEWITTE, P. S., BLINN, T. & PERAKATH, B. (1995) Information Integration for Concurrent Engineering (IICE) IDEF3 Process Description Capture Method Report.
- MCCUTCHEON, D. M. & MEREDITH, J. R. (1993) Conducting case study research in operations management. *Journal of Operations Management*, 11, 239-256.
- MEDVIDOVIC, N. & TAYLOR, R. N. (1997) A framework for classifying and comparing architecture description languages. *Proceedings of the Sixth European Software Engineering Conference, number 1301 in Lecture Notes in Computer Science*. New York, SpringerVerlag.

- MILI, H., FAYAD, M., BRUGALI, D., HAMU, D. & DORI, D. (2002) Enterprise frameworks: issues and research directions. *Software - Practice and Experience*, 32, 801-831.
- MILLER, T. E. & BERGER, D. W. (2001) *Totally Integrated Enterprises*, Boca Raton, St. Lucie Press.
- MORGAN, G. (1996) *Images of Organization*, SAGE Publications.
- MURTHY, R., LIU, Z. H., KRISHNAPRASAD, M. & ET AL (2005) Towards an Enterprise XML Architecture. *SIGMOD 2005*. Baltimore, Maryland, USA, ACM.
- MYERS, M. D. (1997) Qualitative Research in Information Systems, http://www.misq.org/discovery/MISQD_isworld/.
- NEAGA, E. I. & HARDING, J. A. (2005) An enterprise modeling and integration framework based on knowledge discovery and data mining. *International Journal of Production Research*, 43, 1089-1108.
- NEALE, D. C. & CARROLL, J. M. (1997) The Role of Metaphors in User Interface Design. IN HELANDER, M., LANDAUER, T. K. & PRABHU, P. (Eds.) *Handbook of Human-Computer Interaction*. 2nd ed., Elsevier Science B.V.
- NHS QUALITY IMPROVEMENT SCOTLAND (2006) Safe Today - Safer Tomorrow: Patient Safety: Review of Incident and Near-Miss Reporting. p.102.
- NIH (2004) Enterprise Architecture at NIH, Center for Information Technology, National Institutes of Health, <http://datacenter.cit.nih.gov/interface/interface230/ea.html>.
- NORAN, O. (2003) An analysis of the Zachman framework for enterprise architecture from the GERAM perspective. *Annual Reviews in Control*, 27, 163-183.
- OMG (2004) *UML 1.5 Specification: UML Summary*.
- PARIZEAU, Y. (2002) Enterprise Architecture for Complex Government and the Challenge of Government On-Line in Canada. *Faculty of Computer Science*. Halifax, Dalhousie University.
- PARKER, J. (2000) *Structuration*, Buckingham, Open University Press.
- PARNAS, D. L. (1972) On the Criteria to be Used in Decomposing Systems into Modules. *Communications of the ACM*, 15, 1053-1058.
- PAWSON, R. (2000) *Expressive Systems: A manifesto for radical business software*, UK, CSC Research Services.
- PAYNE, S. J. & GREEN, T. T. G. (1986) Task-Action Grammars: A Model of the Mental Representation of Task Languages. *Human-Computer Interaction*, 2, 93-133.
- PIMM, D. (1987) *Speaking mathematically : communication in mathematics classrooms*, London ; New York, Routledge & K. Paul.
- PINTO, H. S. & MARTINS, J. P. (2004) Ontologies: How can They be Built. *Knowledge and Information Systems*, 6, 441-464.
- POLOVINA, S. (1993) The Suitability of Conceptual Graphs in Strategic Management Accountancy. Loughborough University of Technology.
- POOLE, M. S. & DESANCTIS, G. (2003) Structuration Theory in Information Systems Research: Methods and Controversies. IN WHITMAN, M. E. & WOSZCZYNSKI, A. B. (Eds.) *The Handbook for Information Systems Research*. Idea Group Publishing.

- POURABBAS, E. & RAFANELLI, M. (1999) Characterization of Hierarchies and Some Operators in OLAP environment. *DOLAP '99, ACM Second International Workshop on Data Warehousing and OLAP*. Kansas City, Missouri, USA, ACM.
- PRESLEY, A., SARKIS, J., BARNETT, W. & LILES, D. (2001) Engineering the virtual enterprise: An architecture-driven modeling approach. *International Journal of Flexible Manufacturing Systems*, 13, 145-162.
- PROPER, H. A., VERRIJN-STUART, A. A. & HOPPENBROUWERS, S. J. B. A. (2005) On Utility-based Selection of Architecture-Modelling Concepts. *The Second Asia-Pacific Conference on Conceptual Modelling (APCCM2005)*. Newcastle, Australia.
- RAYMOND, K. (2006) Reference Model of Open Distributed Processing (RM-ODP): Introduction, http://www.lcc.uma.es/~av/RM-ODP/Tutorials/ODP_Tutorial-icodp95.pdf.
- ROSTAD, C. C. (2000) Enterprise Modeling for Enterprise Integration: A case of manufacturing. IN ROSTADAS, A. & ANDERSEN, B. (Eds.) *Enterprise Modeling: Improving Global Industrial Competitiveness*. Boston, Kluwer Academic Publishers.
- ROUSSEV, B. & ROUSSEVA, Y. (2004) Software Development: Informing Sciences Perspective. *Issues in Informing Science and Information Technology*, 1, 237-245.
- SANDELANDS, L. E. & STABLEIN, R. E. (1987) The concept of organization mind. *Research in the sociology of organizations*, 135-161.
- SENGE, P. (1990) *The fifth discipline: The art and practice of the learning organization*, London, Century Business.
- SLEMBEK, I. M. (2003) Evaluating and Improving Knowledge-Intensive Work Processes through the Application of Information and Communications Technologies. *Computer Science*. Sydney, University of Technology.
- SMITH, J. M. & SMITH, D. C. P. (1977) Database Abstractions: Aggregation. *Communications of the ACM*, 20, 405-413.
- SOLBERG, C. (2000) Enterprise Modeling and Education. IN ROSTADAS, A. & ANDERSEN, B. (Eds.) *Enterprise Modeling: Improving Global Industrial Competitiveness*. Boston, Kluwer Academic Publishers.
- SOWA, J. F. (2000) *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Pacific Grove, CA, Brooks/Cole.
- STEFIK, M. (1996) *Internet Dreams: Archetypes, Myths, and Metaphors*, London, The MIT Press.
- STUFFLEBEAM, D. L. (2001) Evaluation Models. *New Directions for Evaluation*, Spring 2001, 7-98.
- SUTCLIFFE, A. & PATEL, U. (1996) 3D or not 3D: Is it Nobler in the Mind? IN SASSE, M. A., CUNNINGHAM, R. J. & WINDER, R. L. (Eds.) *People and computers XI: proceedings of HCI '96*. London, Springer.
- SZEGHEO, O. (2000) Introduction to Enterprise Modeling. IN ROSTADAS, A. & ANDERSEN, B. (Eds.) *Enterprise Modeling: Improving Global Industrial Competitiveness*. Boston, Kluwer Academic Publishers.
- SZEGHEO, O. & GASTINGER, A. (2000) Enterprise Modeling Architectures. IN ROSTADAS, A. & ANDERSEN, B. (Eds.) *Enterprise Modeling: Improving Global Industrial Competitiveness*. Kluwer Academic Publishers.

- TAKEUCHI, H. & NONAKA, I. (1986) The New New Product Development Game. *Harvard Business Review*, 137-146.
- THE OPEN GROUP (2002) The Open Group Architectural Framework (TOGAF), Version 8, <http://www.opengroup.org>.
- THEUERKORN, F. (2005) *Lightweight Enterprise Architectures*, Boca Raton, Florida, Auerbach.
- THOMAS, J. J., BOHN, S., BROWN, J. C., PENNOCK, K., SCHUR, A. & WISE, J. A. (1994) Information Visualization: Data Infrastructure Architectures. *Proceedings., Seventh International Working Conference on Scientific and Statistical Database Management*. Charlottesville, VA, USA.
- TP, T. R. (2001) Designing Virtual Worlds as Architecture, citeseer.ist.psu.edu/495173.html.
- TRAVERS, M. D. (1996) Programming with Agents: New metaphors for thinking about computation. *School of Architecture and Planning*, Massachusetts, Massachusetts Institute of Technology.
- TRISTRAM, C. (2001) The next computer interface. *Technology Review*, 104, 52-59.
- UNDERWOOD, J. (1996) Models for Change: Soft Systems Methodology, University of Technology, Sydney, <http://www-staff.mcs.uts.edu.au/~jim/bpt/ssm.html>.
- UNITED STATES DEPARTMENT OF STATE (1999) Information Technology Architecture, http://www.state.gov/www/dept/irm/it_architecture/it_vol1.html.
- UNITED STATES GENERAL ACCOUNTING OFFICE (2003) Information Technology: A Framework for Assessing and Improving Enterprise Architecture Management (Version 1.1), https://secure.cio.noaa.gov/hpcc/docita/files/gao_03_584g_framework_for_%20assessing_and_improving_ea_mgt_ver_1_1_042003.pdf.
- USCHOLD, M., KING, M., MORALEE, S. & ZORGIOS, Y. (1997) The Enterprise Ontology. *Knowledge Engineering Review*, 71-88.
- VAIL III, E. F. (2002) Causal Architecture: Bringing the Zachman Framework to Life. *Information Systems Management*, Summer 2002, 8-19.
- VALLECILLO, A. (2001) RM-ODP: The ISO Reference Model for Open Distributed Processing. *Software Engineering*, 69-99.
- VEASEY, P. W. (2001) Use of enterprise architectures in managing strategic change. *Business Process Management Journal*, 7, 420-436.
- VERNADAT, F. (2002) UEML: towards a unified enterprise modelling language. *International Journal of Production Research*, 40, 4309-4321.
- WALSH, J. P. (1995) Managerial and organizational cognition: Notes from a trip down memory lane. *Organization Science*, 6, 280-321.
- WALSHAM, G. & HAN, C.-K. (1991) Structuration Theory and Information Systems Research. *Journal of Applied Systems Analysis*, 17, 77-85.
- WANG, S. (1999) *Analyzing Business Information Systems: An Object-Oriented Approach*, Boca Raton, Auerbach.
- WARE, C. (2000) *Information Visualization: Perception for Design*, San Francisco, Morgan Kaufman Publishers.
- WAY, E. C. (1991) *Knowledge Representation and Metaphor*, Dordrecht, Kluwer Academic Publishers.
- WEINBERGER, D. (2005) Sorting Data to Suit Yourself. *Harvard Business Review*, 16-18.

- WILLIAMS, T. J. & LI, H. (1998) PERA and GERAM - Enterprise Reference Architectures in Enterprise Integration. IN MILLS, J. & KIMURA, F. (Eds.) *Information Infrastructure Systems for Manufacturing II*. Fort Worth, Texas, Kluwer Academic Publishers.
- WILLIAMSON, K. (2002) *Research methods for students, academics and professionals: Information management and systems*, Wagga Wagga, Centre for Information Studies.
- WINOGRAD, T. & FLORES, F. (1987) *Understanding Computers and Cognition: A New Foundation for Design*, Reading, Massachusetts, Addison-Wesley.
- WOOLDRIDGE, M. (2002) *An Introduction to Multiagent Systems*, West Sussex, John Wiley & Sons, Ltd.
- WORTHEN, B. R., SANDERS, J. R. & FITZPATRICK, J. L. (2003) *Program Evaluation: Alternative Approaches and Practical Guidelines*, Boston, Allyn & Bacon.
- WORTMANN, J. C., HEGGE, H. M. H. & GOOSSENAERTS, J. B. M. (2001) Understanding enterprise modelling from product modelling. *Production Planning and Control*, 12, 234-244.
- ZACHMAN, J. A. (1987) A framework for information systems architecture. *IBM Systems Journal*, 26, 276-292.
- ZACHMAN, J. A. (1997) Enterprise Architecture: The Issue of the Century. *Database Programming and Design*.
- ZACHMAN, J. A. & SOWA, J. (1992) Extending and formalizing the framework for information systems architecture. *IBM Systems Journal*, 31, 590-616.
- ZELM, M. & KOSANKE, K. (2001) A Modelling Language for User Oriented Enterprise Modelling. *3rd Conference Francophone de Modelisation et Simulation "Conception, Analyse et Gestion des Systemes Industriels" MOSIM'01*. Troyes, France.

**10 APPENDIX A – PROJECT SUMMARY FOR
UTS EA PROJECT**

Project Summary for
UTS Enterprise Architecture Project

Project Demographics	Values
Name of Project	UTS Enterprise Architecture Project
Customer Name/Organization	ITD
Primary Customer Interface Person	Ian Waters
Project Start Date	10/08/05
Project Finish Date	14/10/05

Key Roles in Project	Names of People In the Roles
Enterprise Architect	Ian Waters – Senior IT Programs Consultant
Sponsor	Peter James – Acting Director IT Infrastructure & Operations
Key Stakeholder	Peter Demou – Manager, Plans & Programs
Technology Provider	Gerald Khoury – Enterprise Architecture Consultant

Project Objectives	
1	To produce a high-level view of the university's Enterprise Architecture.
2	To show the interrelationships between the different domain architectures.
3	To describe the <i>primary</i> relationships between the targeted systems.
4	To identify the major infrastructure components that support these systems and show the linkages.
5	To identify the major business processes that are supported by the identified application and infrastructure components and show the linkages between them.
Customer Objectives	
1	To produce concise, easily understood, graphical models of the high-level Enterprise Architecture.
2	To develop models that show the interrelationships between business goals and objectives, and IT systems and services.
3	To use the Enterprise Architecture models to identify the impact of change.
Provider Objectives	
1	To create unified Enterprise Architecture models that span heterogeneous ICT domains.
2	To develop an Enterprise Architecture that is concise, easy to expand and modify, and easy to understand.
3	To develop an Enterprise Architecture that is effective as an enterprise planning and evaluation tool.

Milestones	Start Date	End Date	Responsibilities	Notes
Stage 1: Model one system.	10/08/05	19/08/05	IW & GK	
Produce review materials.	15/08/05	19/08/05	IW & GK	Overview of LEAN, UTSONline model, Project Summary.
Review and obtain feedback.	22/08/05	22/08/05	ITIO	ITIO Mgt meeting on 22/08/05
Refine process.	23/08/05	26/08/05	IW & GK	
Refine scope and schedule.	29/08/05	31/08/05	IW & GK	
Stage 2: Model further systems.	01/09/05	30/09/05	IW & GK	Ian Waters back on 27/09/05
Produce review materials.	03/10/05	07/10/05	IW & GK	
Produce recommendations for future action.	03/10/05	07/10/05	IW & GK	
Review and obtain feedback.	10/10/05	10/10/05	ITIO	ITIO Mgt meeting on 10/10/05

Deliverables to be Provided	Review Process	Completion Date
Project plan (this document).	Ian Waters	12/08/05
A set of Enterprise Architecture models that meet the project objectives. These models will be developed using the Lightweight Enterprise Architecture Notation (LEAN).	To be reviewed internally by Peter Demou, Peter James and the ITIO (IT Infrastructure and Operations management group).	03/10/05
Research questionnaires/surveys to determine the efficacy of LEAN to Enterprise Architecture modelling.	Survey to be performed by Gerald Khoury. Results to be published as part of a Doctoral Thesis.	07/10/05
A report on recommendations for future action.	To be reviewed by the ITIO management group.	14/10/05

Special Processes or Practices Used in This Project	Value (if already used)
The Lightweight Enterprise Architecture Notation (LEAN)	
The ITD IT Architecture Framework.	Approved framework that is being trialled within ITD.
Zachman Framework	Used to situate the architectural work products that are produced.

Additional Notes

Stakeholders

The customers of this project are initially ITD and other UTS business system owners, and potentially non-IT business planners.

Project Scope

There are two hundred and eleven UTS identified major systems in total. This project will focus on the modelling of eleven of these systems. These were selected because they are core university business systems with which ITD are heavily involved.

Stages

Stage 1 will focus on modelling the UTSONline system.

Stage 2 will focus on modelling the remaining ten targeted systems.

**11 APPENDIX B – FINAL PROJECT REPORT
FOR UTS EA PROJECT**

ITD

UTS Enterprise Architecture Project

Phase One

PROJECT REPORT

Project Demographics	
Name of Project	UTS Enterprise Architecture Project
Customer Name/Organization	ITD
Primary Customer Interface Person	Ian Waters
Project Start Date	10/08/05
Project Finish Date	10/10/05

Key Stakeholders	
Title	Name
Senior IT Programs Consultant	Ian Waters
Acting Director IT Infrastructure & Operations	Peter James
Manager, Plans & Programs	Peter Demou
Voice and Data Network Manager	Craig Laughton
Technical Implementation Manager	Sean Donovan
Information Systems Manager	Emily Latif
Applications Project Manager	Pat Player
Enterprise Architecture Consultant	Gerald Khoury

Introduction

ITD has commenced the task of developing a UTS Enterprise Architecture (EA). An EA is a holistic set of models that represent an enterprise's information systems in order to manage change. This document presents the results of the latest stage of this project, which is aimed at developing a set of high-level, holistic, graphical models that tie together disparate, but strategically important components of the UTS IT environment.

In reality, the development of an EA is never complete: business needs, and technological solutions, are always changing. The development of an EA is a means by which that change can be managed, and by which the power of change can be harnessed.

EA's are growing in importance as tools for managing change within today's highly dynamic, demand driven and highly competitive business environments. As the rate of technological change increases and information environments become more complex, more sophisticated methods are needed to manage these environments effectively. EA's help manage this change and overcome the problems of building isolated IT solutions that fail to support an enterprise's vision, goals and objectives. It is for these reasons that the development of EA's is now high on the agenda of most leading organisations.

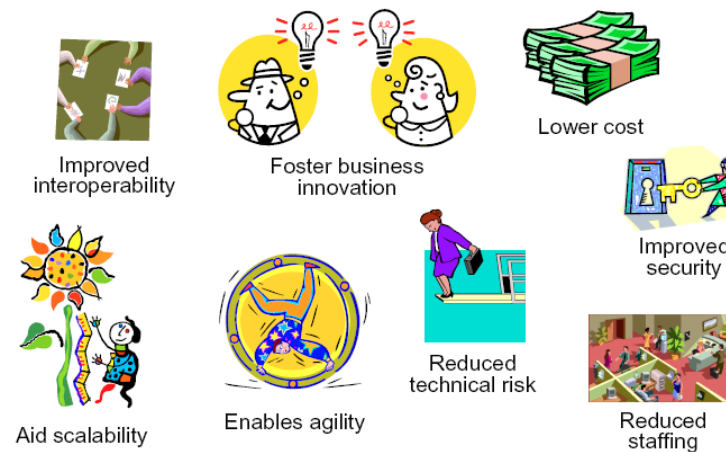


Figure 1: EA's help to deal with the increasing complexity, information overload and demands for higher quality, timeliness and effectiveness.

Project Goals

The objectives of this project were defined as follows:

1. To produce a high-level view of the University's Enterprise Architecture (EA).
2. To show the interrelationships between the different domain architectures.
3. To describe the *primary* relationships between the targeted systems.
4. To identify the major infrastructure components that support these systems and show the linkages.
5. To identify the major business processes that are supported by the identified application and infrastructure components and show the linkages between them.

The customer objectives were defined as:

1. To produce concise, easily understood, graphical models of the high-level EA.
2. To develop models that show the interrelationships between business goals and objectives, and IT systems and services.
3. To use the EA models to identify the impact of change.

Provider objectives were defined as:

1. To create unified EA models that span heterogeneous ICT domains.
2. To develop an EA that is concise, easy to expand and modify, and easy to understand.
3. To develop an EA that is effective as an enterprise planning and evaluation tool.

Project Methods

This project was executed using input from a number of key stakeholders who provided expert knowledge on different aspects of the University's IT environment.

The EA is being modelled using a new modelling language developed by researchers at UTS: The Lightweight Enterprise Architecture Notation (LEAN). The primary advantage of this modelling language over other approaches is that it is a *unified* language that can describe multiple IT domains using a simple, graphical notation. LEAN is designed to be:

- Agile
- Easy to use
- Collaborative
- Thought provoking

Project Scope

More than two hundred and eleven major systems have been identified across the University. In order to achieve a high quality outcome in a relatively short period, a small subset of these systems was selected for inclusion as part of this project. These were selected because they are core university business systems with which ITD are heavily involved. Table 1 lists the applications and systems that were modelled as part of this project.

	Application	System Name
1	Online Learning	UTSOnline
2	Finance	neo
3	Human Resources	neo
4	Student	CASS
5	Timetabling	SYLLABUS+
6	Room Allocation	ALLOCATE+
7	Email	Email System
8	Web Server	Web Server
9	Network	Network
10	Identity Management	LDAP

Table 1: In-Scope Applications

Project Results

The development of this project has provided a set of high-level, holistic models of key UTS systems. Even during the development of this project, it has been found that these models have provided the following benefits:

- The models have aided understanding of the current environment.
- The models have generated discussion about various aspects of the current environment, leading to new insights.
- The models have engendered ideas for simplification or augmentation of the current environment.

It is expected that these benefits will continue to expand as the models are distributed throughout the ITD and user community.

ITD now has the technology and skills to develop further models of the UTS IT environment using LEAN. The use of LEAN provides ITD with a unique advantage in developing easy to use, holistic EA models that support its strategic IT planning and development activities.

Recommendations

The following recommendations are presented for the consideration of ITD management.

Technical Recommendations

As part of the overall program to develop and maintain an ITD EA, the following project activities are recommended. These are summarised as follows and detailed further below:

1. Extend the LEAN models to cover additional systems.
2. Augment the LEAN models to provide more detail.
3. Develop a web-based, EA knowledge management system, within which the LEAN models form one component.

Recommendation 1

In this project, a small but key subset of ITD managed systems was selected for modelling. This achieved two goals. Firstly, it acted as a proof-of-concept of the LEAN modelling approach. Secondly, it meant that the project could be delivered quickly and efficiently. With these goals accomplished, it makes sense to extend the coverage of LEAN models to include additional ITD managed systems and applications. As the purpose of these EA models is to provide support for technology planning and change management, more extensive coverage will lead to a more than proportionate increase in utilisation benefits.

Recommendation 2

The 'first pass' in modelling the chosen systems was deliberately kept at a very high-level. By adding more depth to these models, they can be used to understand UTS systems in greater detail. For example, we could drill down into an identified system resource to learn more about its database schemas, server topology and network interfaces. Once an additional level of detail has been added, the LEAN models can be connected to domain-specific models (eg. a UML application model or an Entity-Relationship data model) providing a seamless flow between high-level plans and more detailed architecture models.

Recommendation 3

Finally, providing a simple, but effective, *EA knowledge management system* can significantly enrich the value of ITD's EA activities. The models developed as part of this project would form an integral part of this system, and would be linked into other existing, and to-be-developed architectural assets. The system would ideally be developed as a web-based system (with appropriate access control) as part of the current intranet environment. This would provide several benefits.

Firstly, as a 'central point of access' it would allow all EA stakeholders and users to easily access needed EA information from any location. This would increase the utilisation of intellectual capital assets that are produced as part of the EA program and ensure that maximum value is achieved from any such product by ensuring it is always available when needed to support ITD program activities.

Secondly, by acting as a single repository for EA deliverables, the EA knowledge management system would ensure that valuable intellectual capital assets are neither lost, duplicated, nor end up as 'shelf ware' due to a lack of awareness and exposure.

And lastly, the web-based interface would provide very user-friendly methods for *navigating* the EA environment by visually displaying the interconnections between various components and providing graphical features that allow users to drill-down into any component to reveal further detail. This is a proven approach to EA management and a demonstration of a previously built commercial system is available upon request.

Organisational Recommendations

The development and management of an EA is now a well accepted and key strategy that is used by the vast majority of leading enterprises to leverage maximum potential from their IT investments. In order to achieve this, most large companies have dedicated resources that fulfil the functions of EA development and governance.

Without a formal EA function, decisions about technology planning and deployment tend to be devolved to individual business groups, with the result that objectives tend to be shorter term, and advantages more localised. The *centralisation* of this strategic architecture function ensures that greater economies of scale can be achieved with less redundancy and greater overall flexibility. This puts UTS in a better position to manage change and exploit new technological opportunities.

In order to achieve these outcomes, it is recommended that ITD set up an EA office that is staffed by dedicated EA resources. These enterprise architects will develop a formal program for EA activities, develop and maintain the ITD EA, and manage the day-to-day governance of EA related activities. This ensures that the Corporate IT Strategy supports the UTS Corporate Strategy and will provide the Branches with the necessary support to ensure that maximum gain is made from UTS IT investments.

Appendix A – ITD System Models

The ITD system models were developed using the LEAN notation. This section shows the nodes and relationships that can be used to develop LEAN models, followed by the actual models that were produced.

Key to LEAN Nodes

Figure 2 shows the all of the node types that can be used in LEAN models.

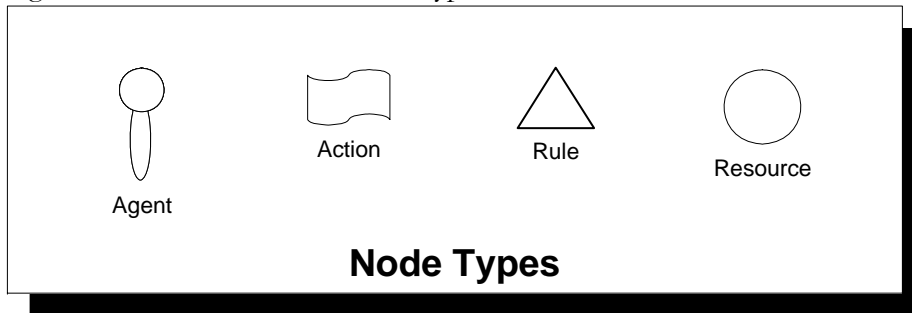


Figure 2 - LEAN Node Types

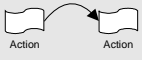






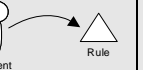

The LEAN nodes are formally defined in Appendix B – LEAN Node Definitions.

Key to LEAN Relationships

Table 2 shows the generic LEAN relationship set. This shows all of the possible relationships that can be used to connect any pair of nodes.

Architectural Models

Pages 13 to 27 present the models that have been produced as part of this project.

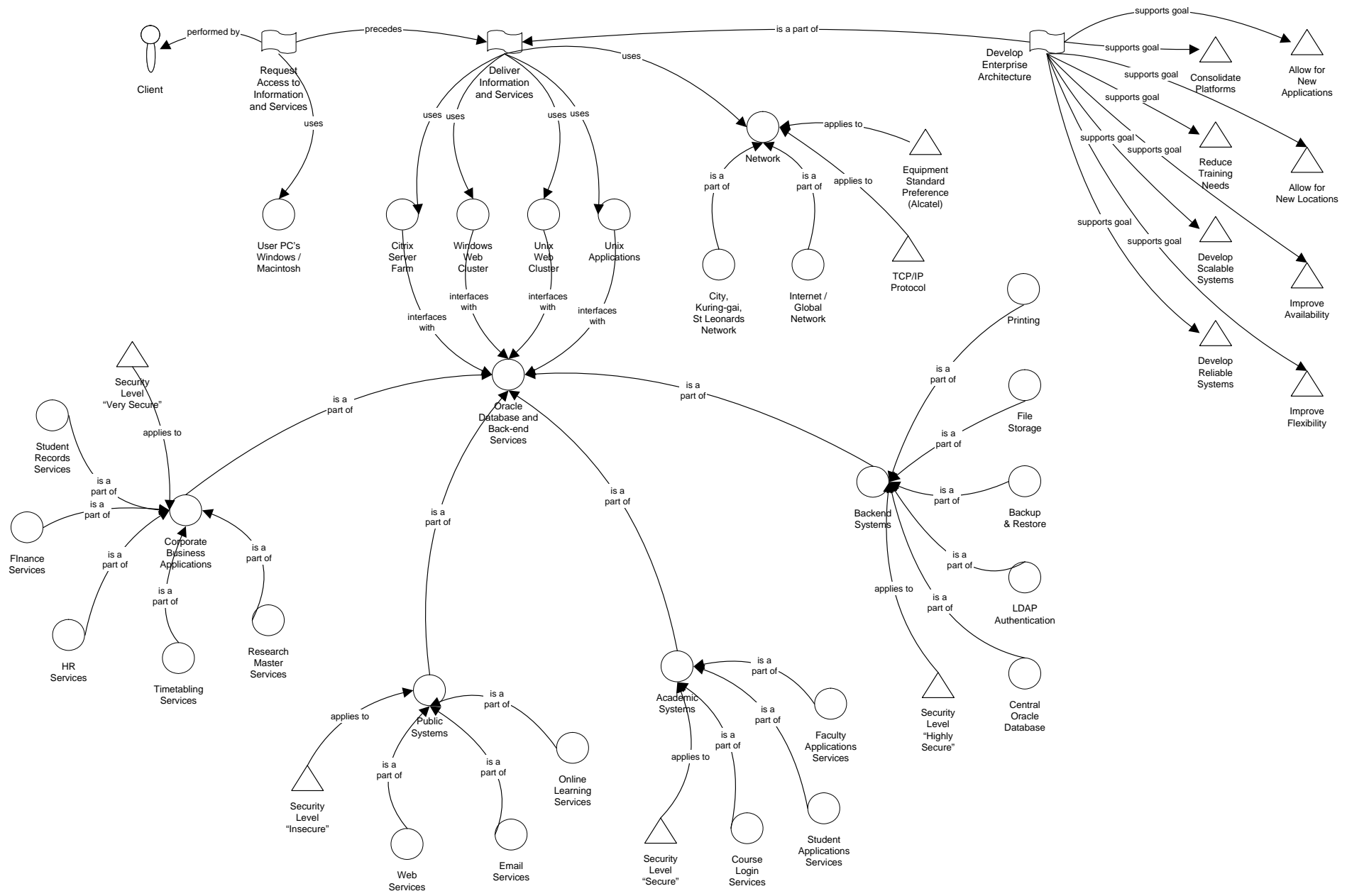
		LEAN NODE PAIRINGS									
		HOMOGENOUS PAIRINGS					HETEROGENOUS PAIRINGS				
RELATIONSHIP SET											
<i>is a type of</i>		✓	✓	✓	✓	✗	✗	✗	✗	✗	
<i>supports</i>		✗	✗	✓	✗	✗	✗	✗	✗	✗	
<i>interfaces with</i>		✗	✗	✓	✗	✗	✗	✗	✗	✗	
<i>is a part of</i>		✓	✓	✓	✓	✗	✗	✗	✗	✗	
<i>precedes</i>		✓	✗	✗	✗	✗	✗	✗	✗	✗	
<i>reports to</i>		✗	✓	✗	✗	✗	✗	✗	✗	✗	
<i>performed by</i>		✗	✗	✗	✗	✓	✗	✗	✗	✗	
<i>uses</i>		✗	✗	✗	✗	✗	✓	✗	✗	✗	
<i>produces</i>		✗	✗	✗	✗	✗	✓	✗	✗	✗	
<i>complies with</i>		✗	✗	✗	✗	✗	✗	✓	✓	✓	
<i>has applicable</i>		✗	✗	✗	✗	✗	✗	✓	✓	✓	
<i>supports goal</i>		✗	✗	✗	✗	✗	✗	✓	✗	✗	

✓ = relationship allowed. ✗ = relationship not allowed.

Table 2 - Generic LEAN Relationship Set

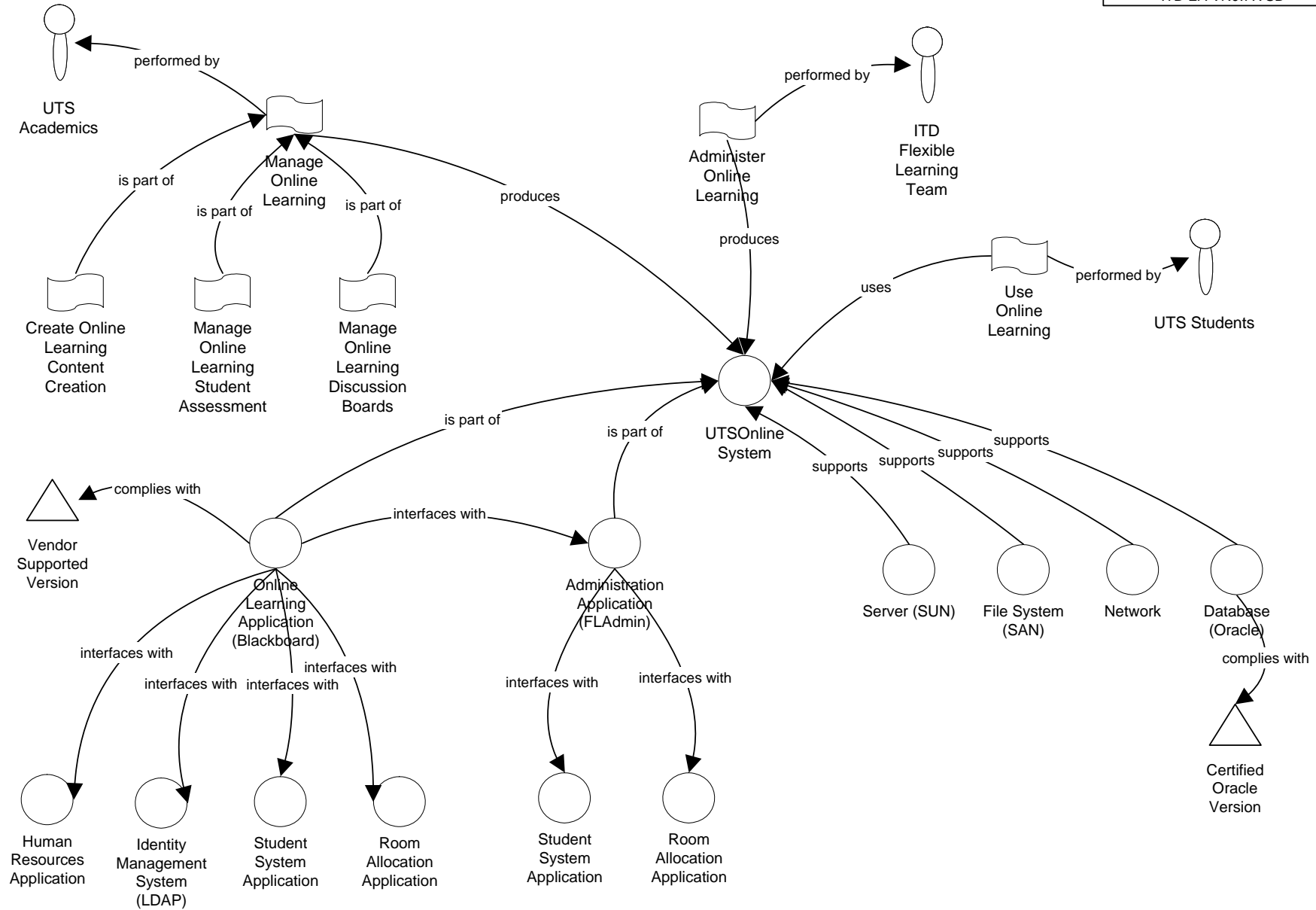
System	Application	View
Technical Infrastructure	All Applications	Universal

Owner
Filename
ITD EA VN0.7.VSD



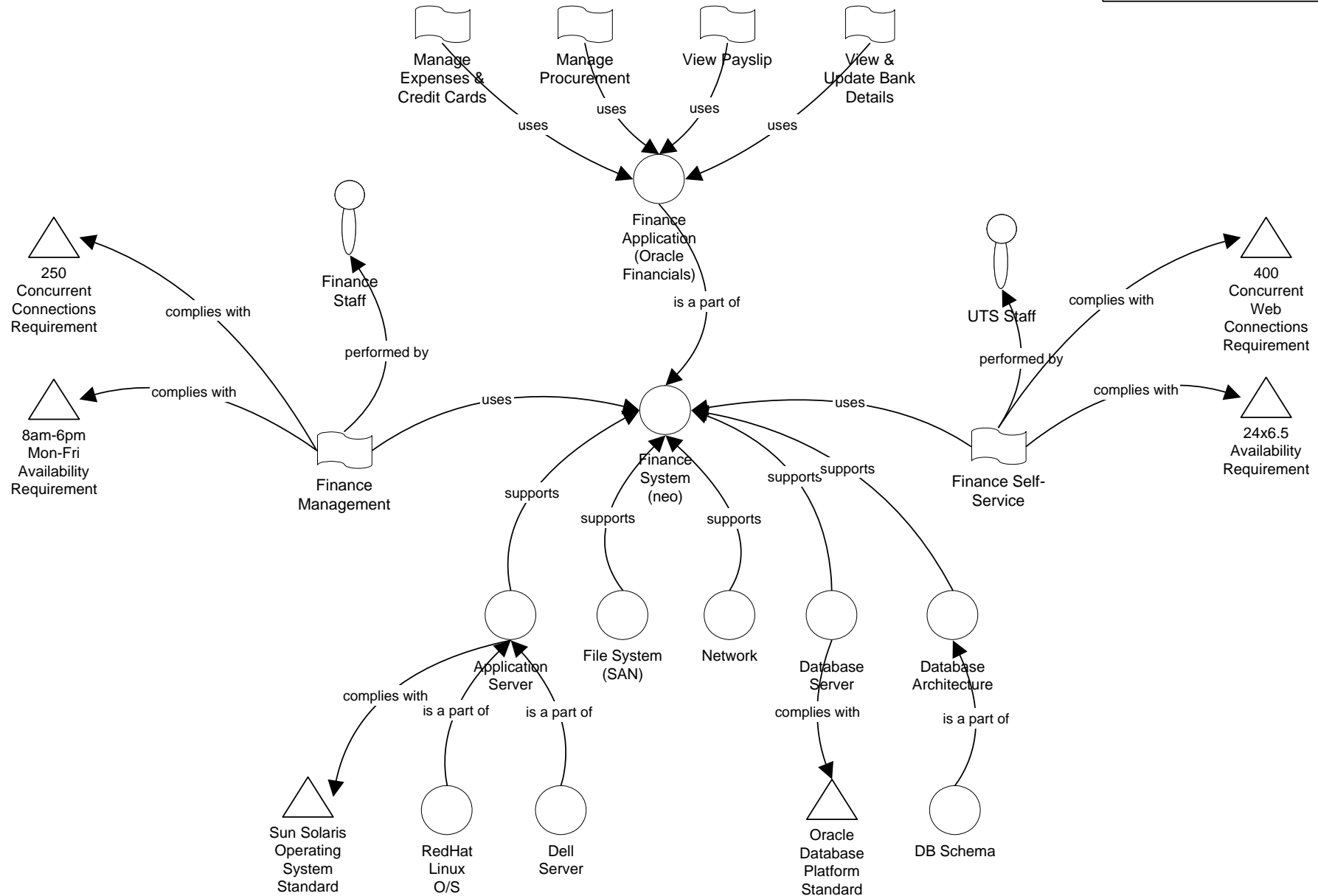
System	Application	View
UTSOnline	Blackboard & FLAdmin	Universal

Owner
Filename
ITD EA VN0.7.VSD



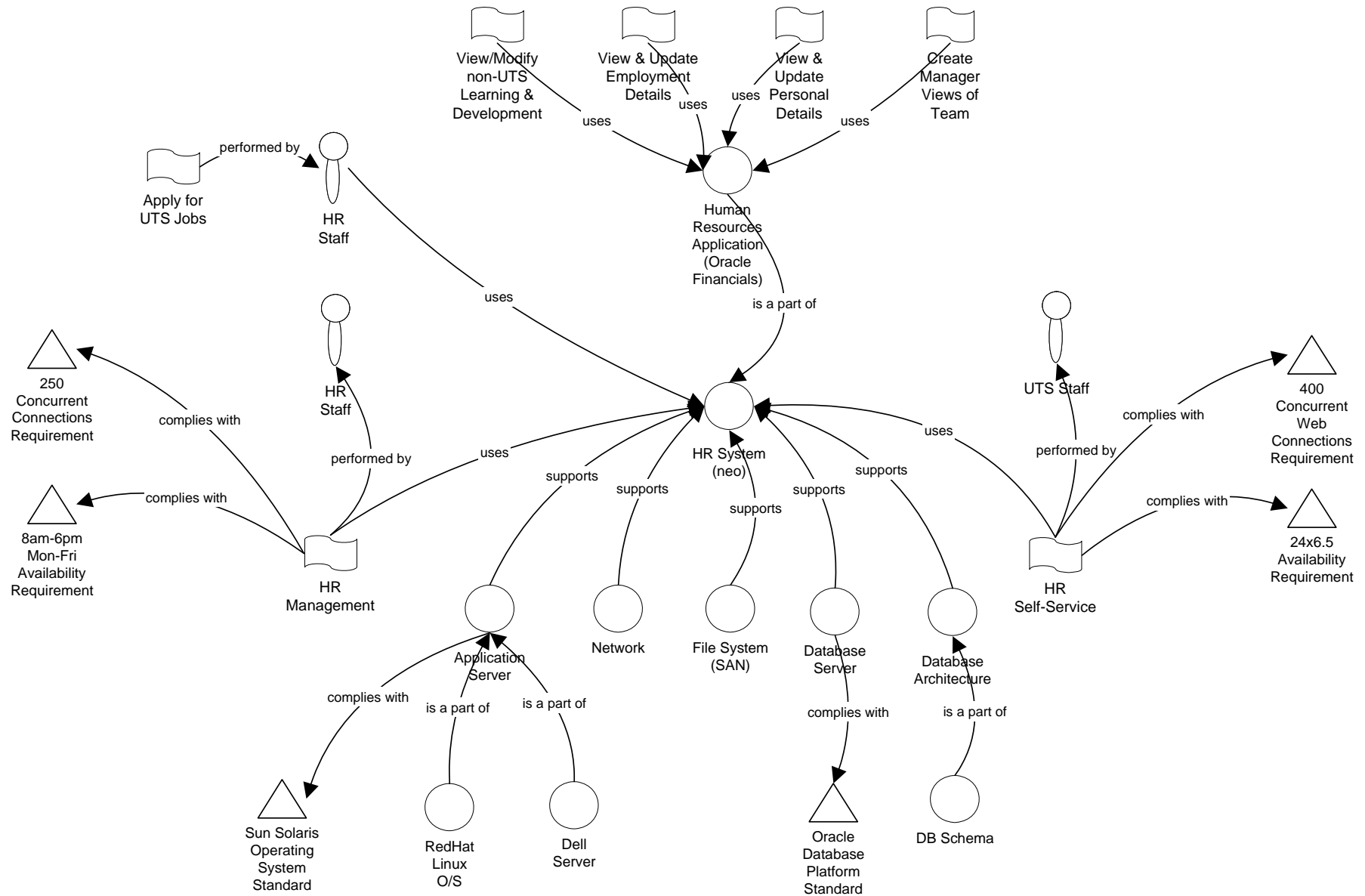
System	Application	View
neo	Finance	Universal

Owner
Filename
ITD EA VN0.7.VSD



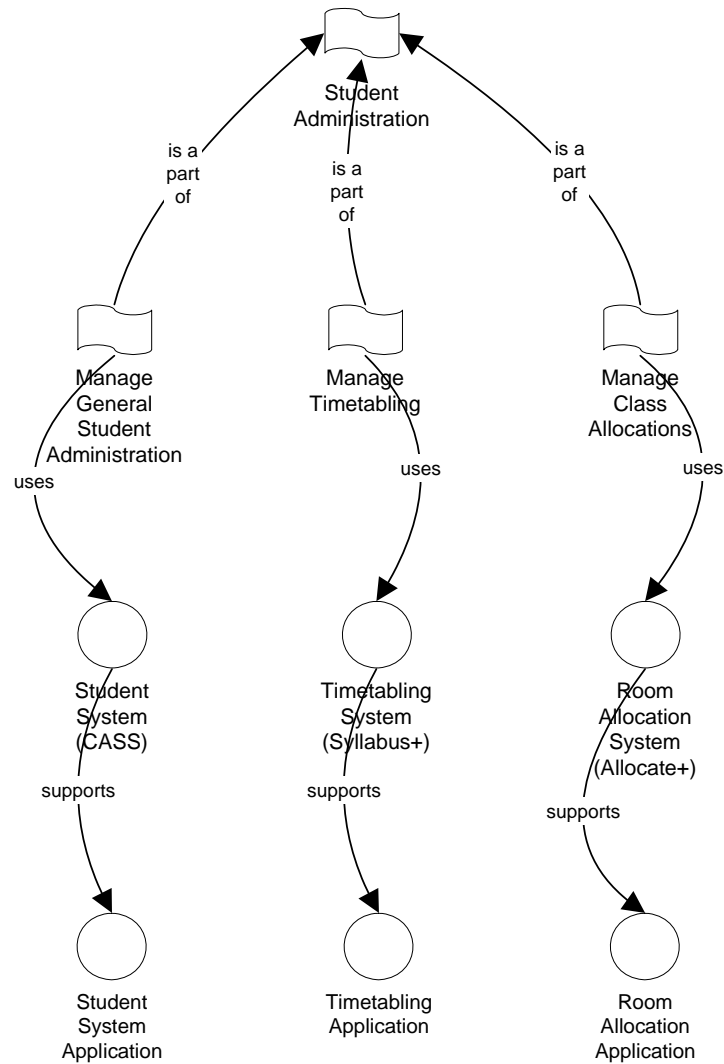
System	Application	View
neo	HumanResources	Universal

Owner
Filename
ITD EA VN0.7.VSD



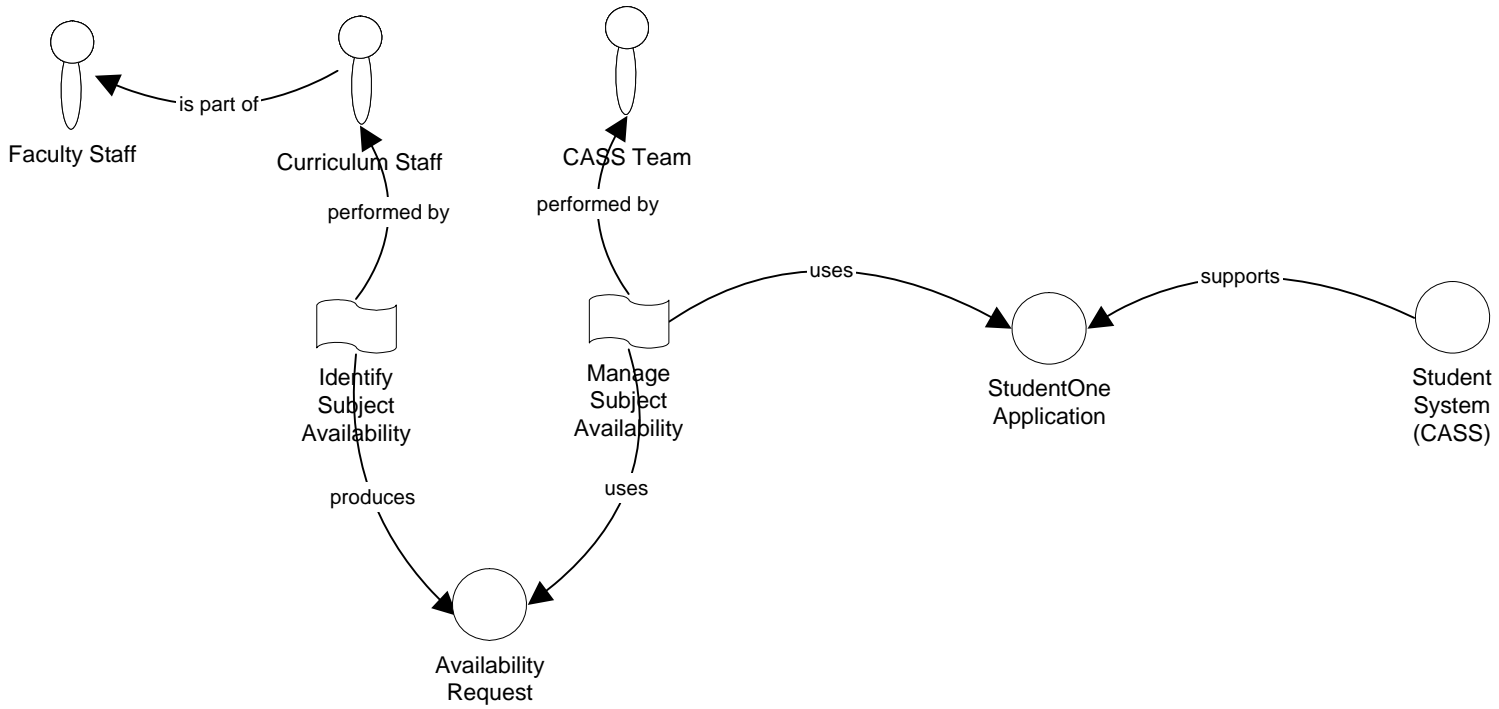
System	Application	View
Student Admin. Systems	Student Admin. Applications	Universal

Owner
Filename ITD EA VN0.7.VSD



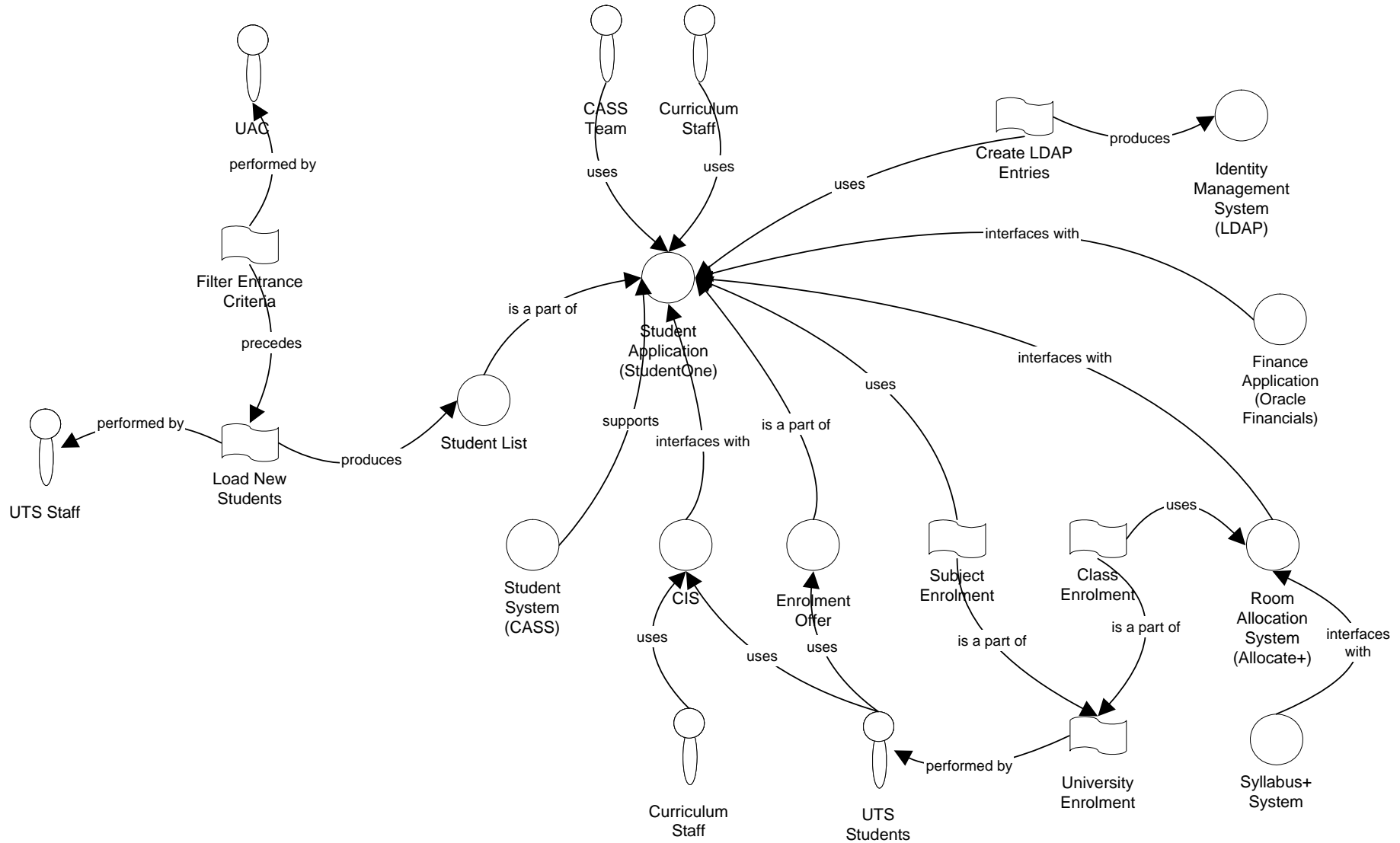
System	Application	View
CASS	StudentOne	Subject Mgt

Owner
Filename
ITD EA VN0.7.VSD



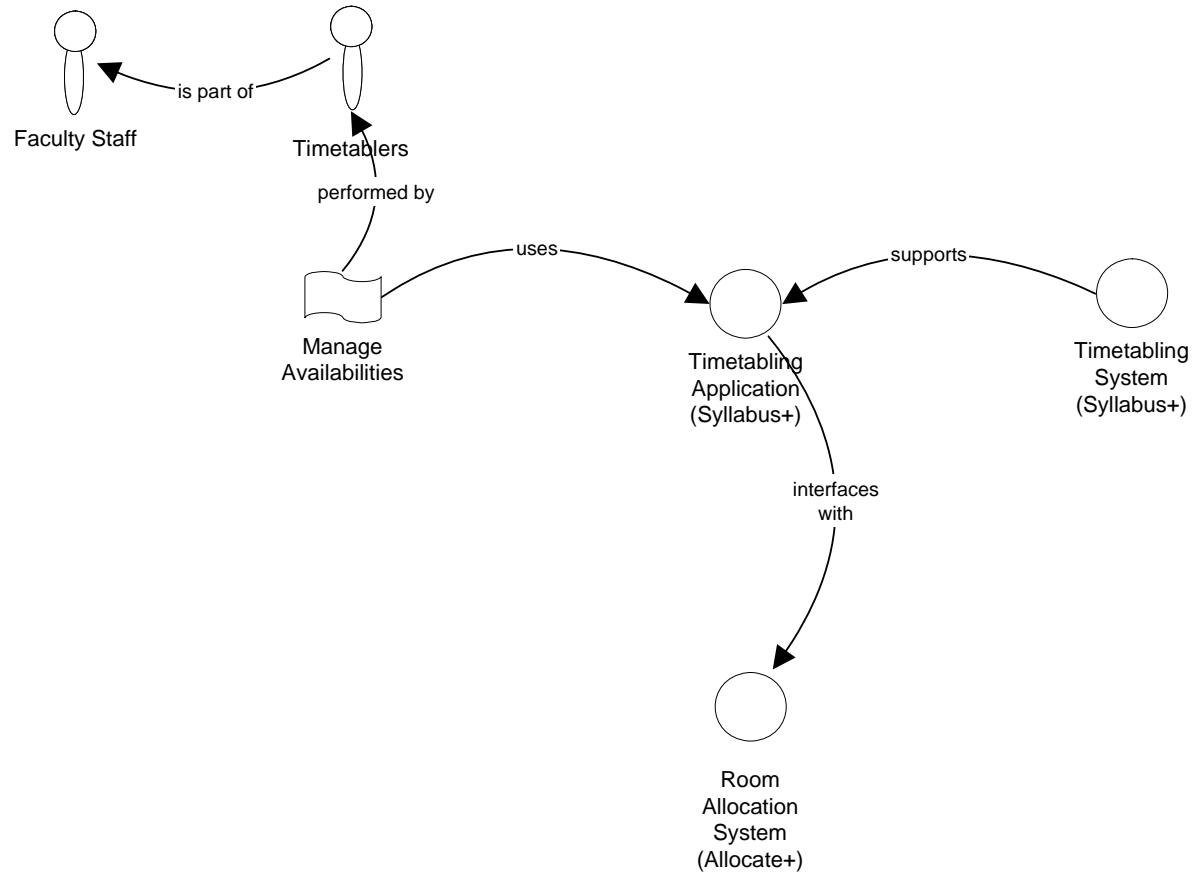
System	Application	View
CASS	StudentOne	Enrolments

Owner
Filename
ITD EA VN0.7.VSD



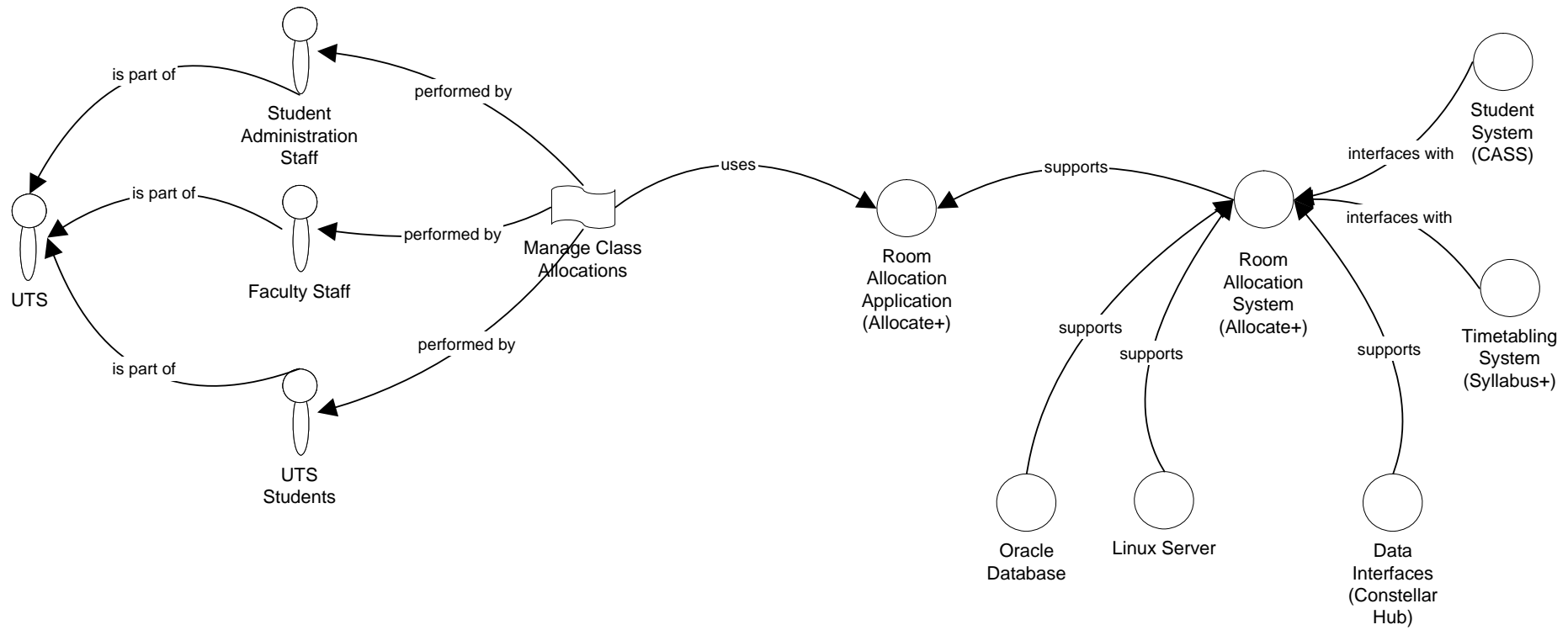
System SYLLABUS+	Application Timetabling	View Universal
---------------------	----------------------------	-------------------

Owner
Filename ITD EA VN0.7.VSD



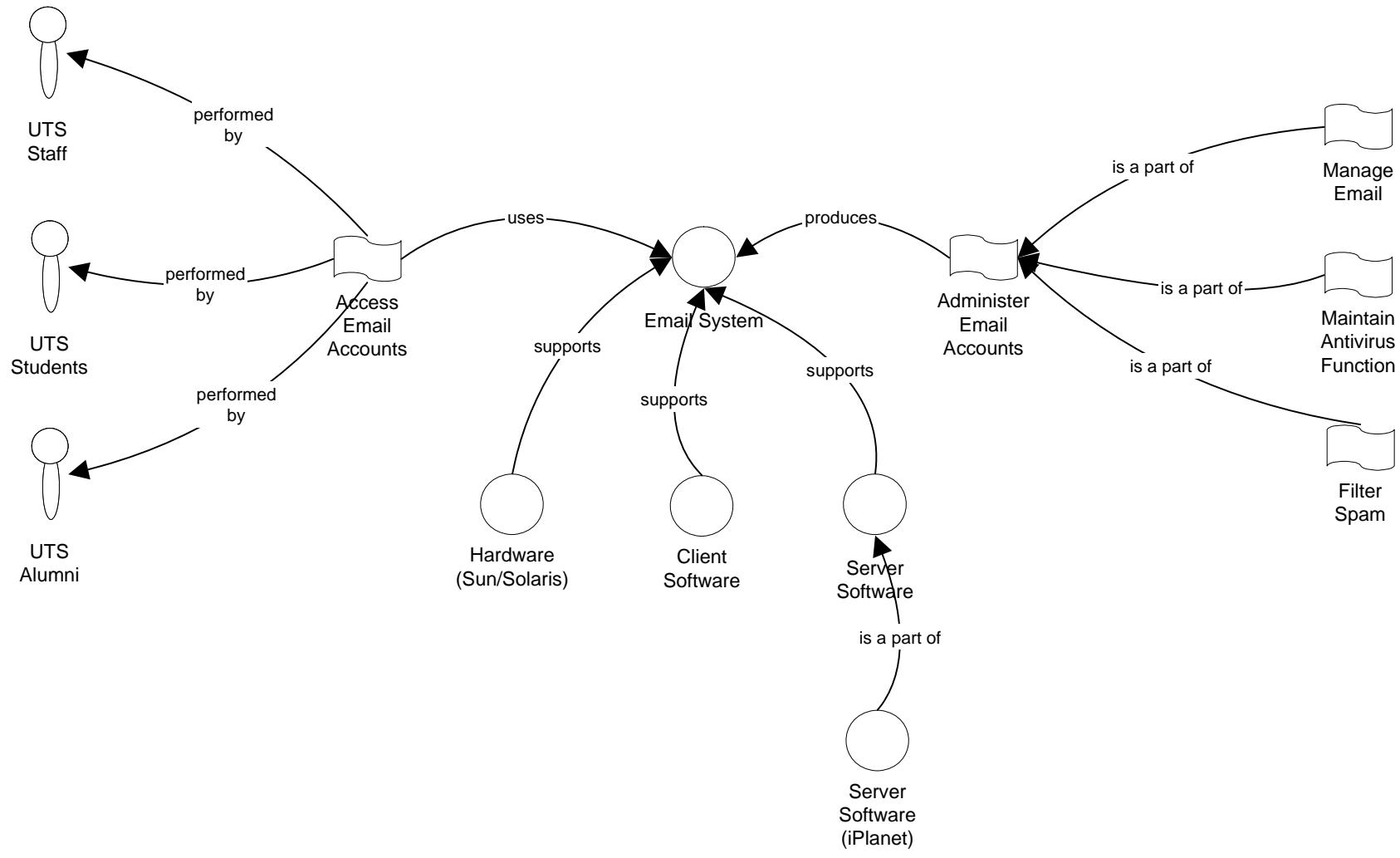
System	Application	View
ALLOCATE+	Room Allocation	Universal

Owner
Filename
ITD EA VN0.7.VSD



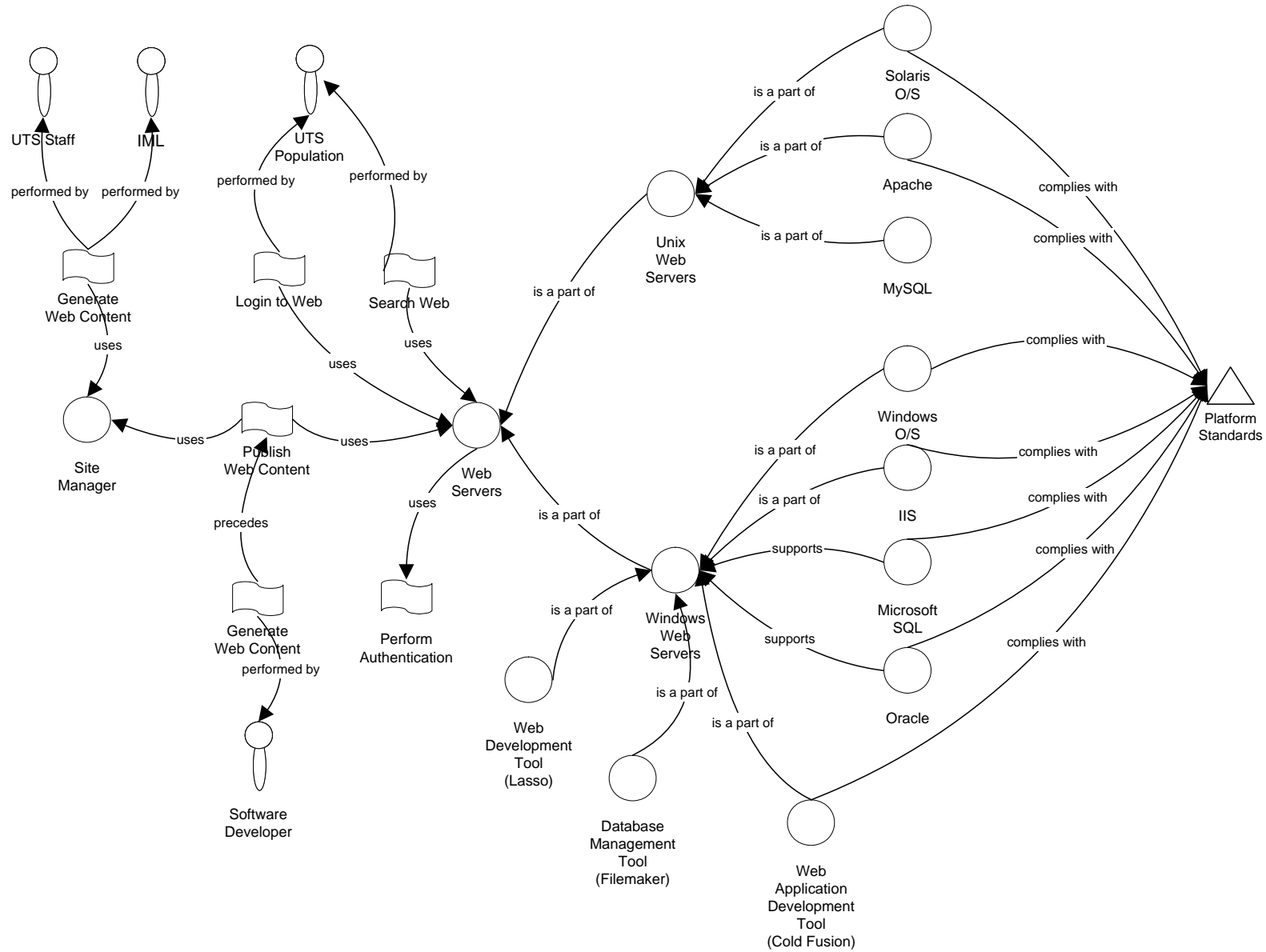
System	Application	View
Email System	Email	Universal

Owner
Filename
ITD EA VN0.7.VSD



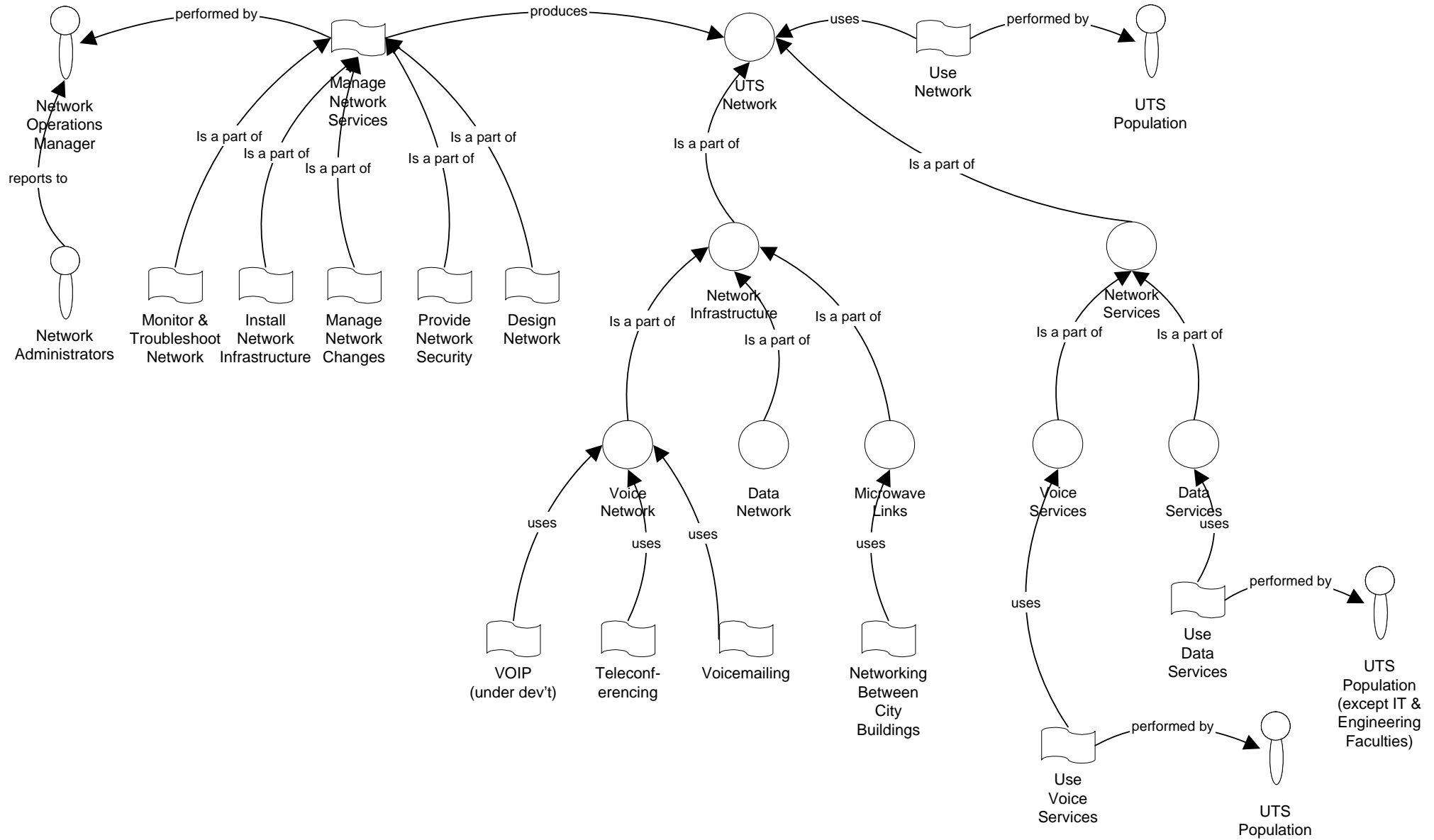
System	Application	View
Web Server	Web Server	Universal

Owner
Filename
ITD EA VN0.7.VSD



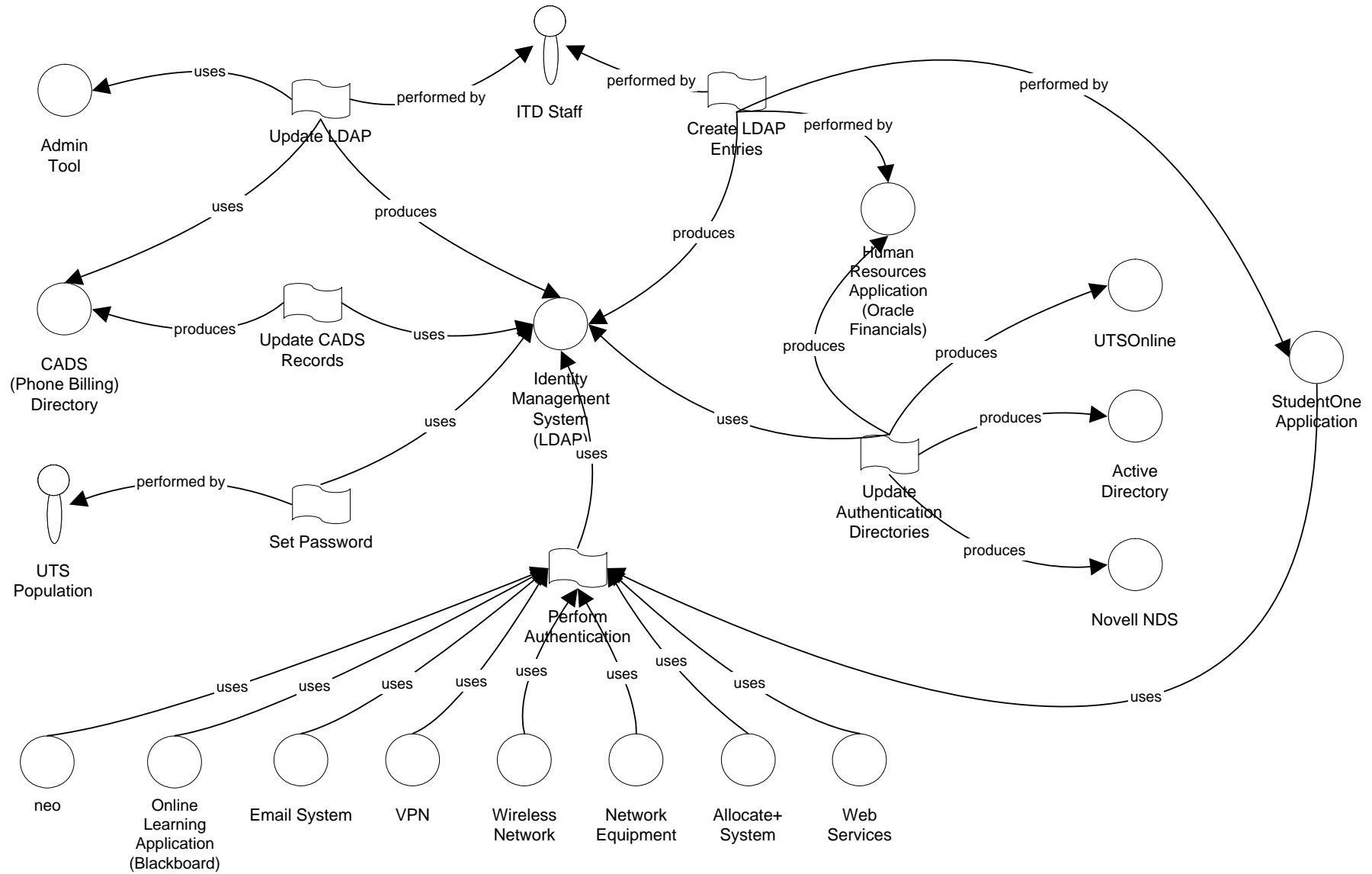
System	Application	View
Network	N/A	Universal

Owner
Filename
ITD EA VN0.7.VSD



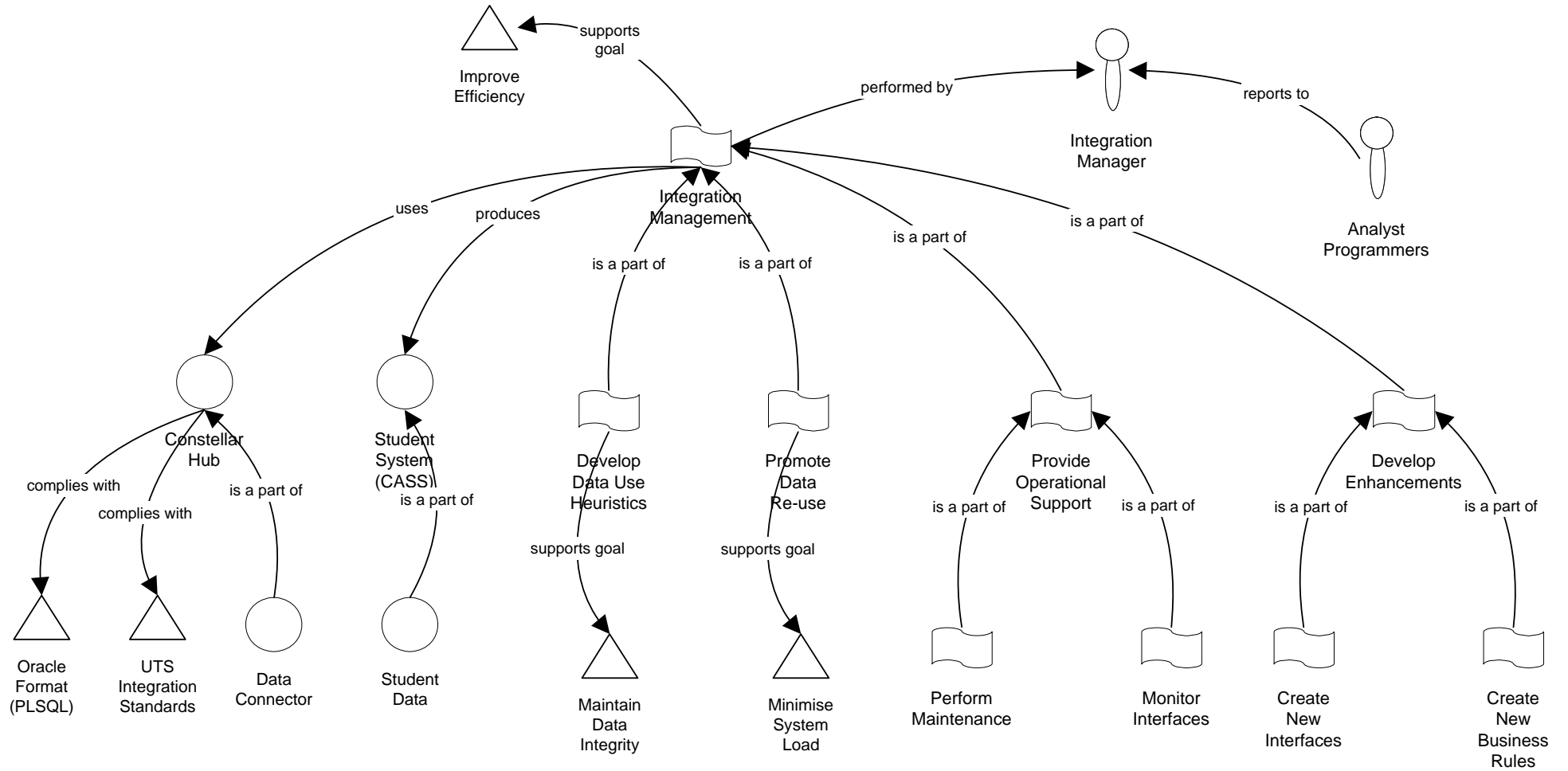
System	Application	View
LDAP	Multiple	Universal

Owner
Filename
ITD EA VN0.7.VSD



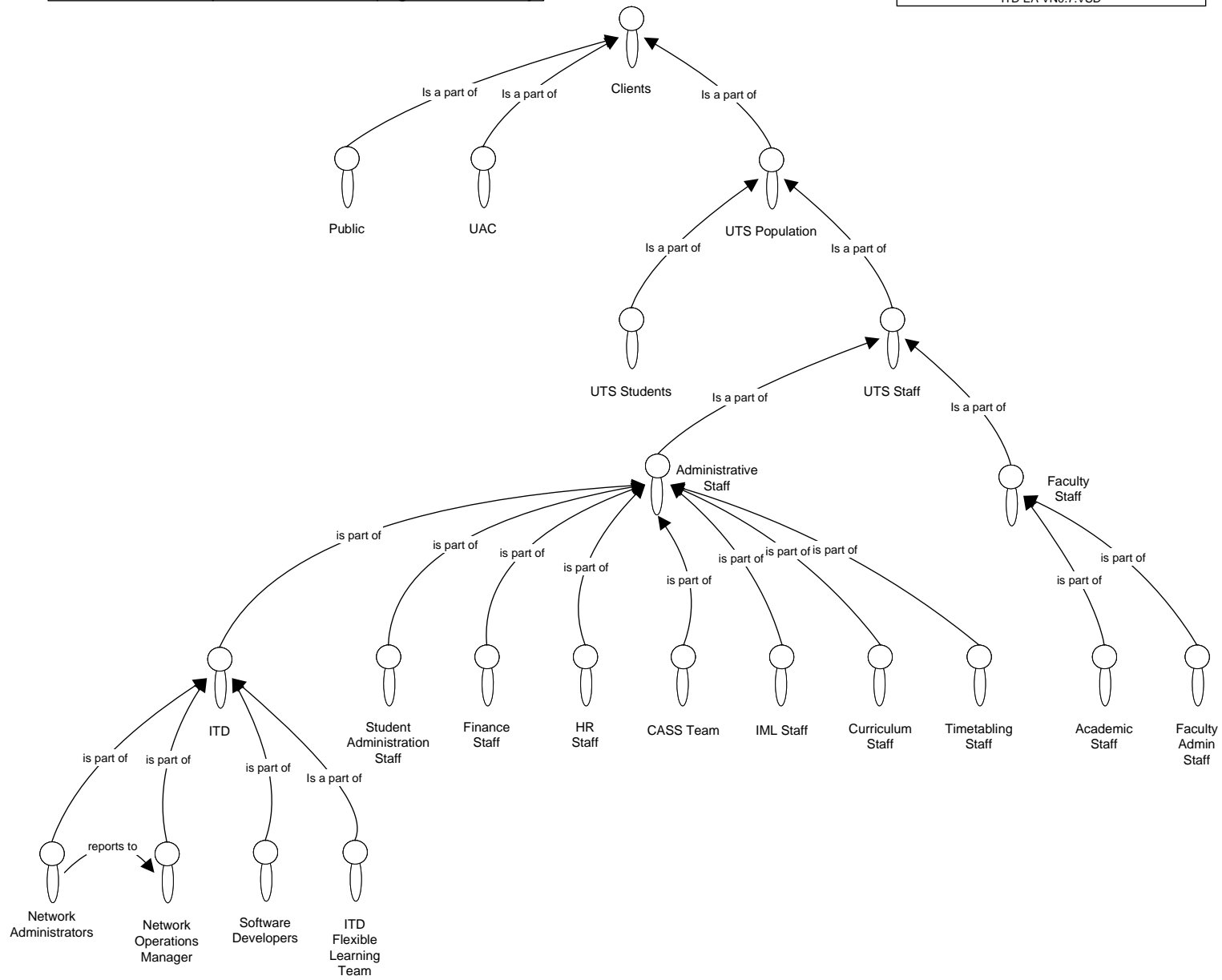
System	Application	View
Integration	Multiple	Universal

Owner
Filename ITD EA VN0.7.VSD



System	Application	View
N/A	N/A	Agent Taxonomy

REV.	DESCRIPTION	DATE	BY
FILENAME	ITD EA VN0.7.VSD		



Appendix B – LEAN Node Definitions

There are four node types used in LEAN:

- Agent
- Action
- Rule
- Resource

These are termed ‘universal’ types, since ‘non-universal’ types can also be represented as nodes. Non-universal types are subtypes of the universal types. Non-universal types are explained in more detail in section 6.5 LEAN Type Hierarchies.

The sections below describe each of the four node types. The labels shown on the graphical representations of each type are the defaults. That is, if no label is provided, the node is assumed to be of the universal type.

Agents

Definition

An Agent is an entity that can exert power in order to produce an effect. In relation to IT systems, the immediate effect is the exchange of information.

Description

In LEAN, the effects produced by Agents are referred to as 'Actions'.

Graphical Representation

The Agent node is represented using the following icon:



Examples

Agents may be:

- People
- Roles
- Organisations
- Communities
- Nation-states
- Systems

Notes

In the case of temporal events, the Agent may be the system itself. In all other cases, the Agent is the entity that triggers a system event.

Resources

Definition

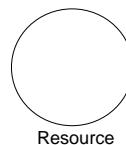
A Resource is a structured property of the modelled system that can be consumed or produced by one or more Agents.

Description

A Resource represents a natural constraint within the system.

Graphical Representation

The Resource node is represented using the following icon:



Examples

Resources may be:

- Raw materials

- Systems

- Documents
- Images
- Services
- Agents

Notes

When Agents are used to signify constraints on the system, they can be represented as Resources.

Rules

Definition

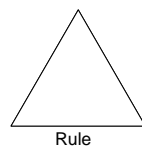
A Rule defines a sanctioned mode of conduct.

Description

A Rule regulates the type of Actions that may take place within a system.

Graphical Representation

The Rule node is represented using the following icon:



Examples

- Physical constraints

- Logical constraints

- Legal and regulatory compliance

- Standards and guidelines

- Business goals or objectives

Notes

Actions

Definition

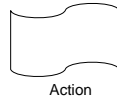
An Action is an activity that is performed. Actions equate to the *capabilities* that Agents possess.

Description

Agents, Rules and Resources can only interact with each other through Actions.

Graphical Representation

The Action node is represented using the following icon:



Examples

- Addition, modification or deletion of data, information or systems
- Identification or selection of data, information or systems

Notes

Many modelling languages identify the concept of an event that triggers some action. In fact, an event can simply be viewed as an action that is performed by another agent and modelled this way in LEAN.

Appendix C – References

In addition to expert input by ITD staff, the following documents were used as reference materials for this project:

- Project Summary for UTS Enterprise Architecture Project Vn.0.2
- ITD UTS Enterprise Architecture Project Stage II Vn.0.1
- ITD UTS Enterprise Architecture Project Modelling Standards Vn.0.1
- ITD Systems Architecture – July 2002
- Information Technology Division IT Architecture – Suggested Standards
- Information Technology Division IT Architecture – Suggested Domain Structure
- Information Technology Division IT Architecture – Documentation of Standards
- Information Technology Division IT Architecture Framework Vn.0.53
- UTS Corporate IT Architecture (Layers Model)
- Student One Conceptual Architecture Diagram – February 2002
- UTS Online Components for Risk Assessment

- UTS Technical Hardware and Software Architecture for the Implementation of the Oracle Financial and Human Resources System Vn. 5 – October 2003
- UTS Technical Hardware and Software Architecture for the Implementation of the Student One Student Records System Vn. 3
- Allocate+ Components for Risk Assessment – 20 May 04
- CASS and Syllabus+ Availabilities
- Finance and Human Resources System Infrastructure Schematic - March 2003
- University of Technology Network Domain Architecture – Draft
- Request for Proposal on Storage and Backup Systems for UTS Email Project
- Email Project Implementation Overview Plan – October 26, 1998
- Implementation Plan Email Project – 17 February 1999

12 APPENDIX C – QUESTIONNAIRES

RESULTS QUESTIONNAIRRE - CLOSED QUESTIONS		
BUSINESS USERS		
#	QUESTION	RESPONSE
1	I found LEAN easy to use.	<input type="radio"/> 0-don't know <input type="radio"/> 1-very strongly agree <input type="radio"/> 2-strongly agree <input type="radio"/> 3-agree <input type="radio"/> 4-disagree <input type="radio"/> 5-strongly disagree <input type="radio"/> 6-very strongly disagree
2	LEAN is an effective language for modelling high-level (conceptual) information.	<input type="radio"/> 0-don't know <input type="radio"/> 1-very strongly agree <input type="radio"/> 2-strongly agree <input type="radio"/> 3-agree <input type="radio"/> 4-disagree <input type="radio"/> 5-strongly disagree <input type="radio"/> 6-very strongly disagree
3	LEAN is an effective language for modelling low-level (detailed) information.	<input type="radio"/> 0-don't know <input type="radio"/> 1-very strongly agree <input type="radio"/> 2-strongly agree <input type="radio"/> 3-agree <input type="radio"/> 4-disagree <input type="radio"/> 5-strongly disagree <input type="radio"/> 6-very strongly disagree

4	LEAN captures information across all technical domains of interest.	<input type="radio"/> 0-don't know <input type="radio"/> 1-very strongly agree <input type="radio"/> 2-strongly agree <input type="radio"/> 3-agree <input type="radio"/> 4-disagree <input type="radio"/> 5-strongly disagree <input type="radio"/> 6-very strongly disagree
5	LEAN captures information from all business areas of interest.	<input type="radio"/> 0-don't know <input type="radio"/> 1-very strongly agree <input type="radio"/> 2-strongly agree <input type="radio"/> 3-agree <input type="radio"/> 4-disagree <input type="radio"/> 5-strongly disagree <input type="radio"/> 6-very strongly disagree
6	LEAN leads users to think more deeply about the structures and relationships that exist.	<input type="radio"/> 0-don't know <input type="radio"/> 1-very strongly agree <input type="radio"/> 2-strongly agree <input type="radio"/> 3-agree <input type="radio"/> 4-disagree <input type="radio"/> 5-strongly disagree <input type="radio"/> 6-very strongly disagree
7	LEAN models convey more meaning than the models I previously used.	<input type="radio"/> 0-don't know <input type="radio"/> 1-very strongly agree <input type="radio"/> 2-strongly agree <input type="radio"/> 3-agree <input type="radio"/> 4-disagree <input type="radio"/> 5-strongly disagree <input type="radio"/> 6-very strongly disagree

8	LEAN models convey meaning more precisely than the models I previously used.	<input type="radio"/> 0-don't know <input type="radio"/> 1-very strongly agree <input type="radio"/> 2-strongly agree <input type="radio"/> 3-agree <input type="radio"/> 4-disagree <input type="radio"/> 5-strongly disagree <input type="radio"/> 6-very strongly disagree
9	I would use LEAN again for enterprise architecture modelling.	<input type="radio"/> 0-don't know <input type="radio"/> 1-very strongly agree <input type="radio"/> 2-strongly agree <input type="radio"/> 3-agree <input type="radio"/> 4-disagree <input type="radio"/> 5-strongly disagree <input type="radio"/> 6-very strongly disagree
10	I would recommend LEAN for use by other enterprise architects.	<input type="radio"/> 0-don't know <input type="radio"/> 1-very strongly agree <input type="radio"/> 2-strongly agree <input type="radio"/> 3-agree <input type="radio"/> 4-disagree <input type="radio"/> 5-strongly disagree <input type="radio"/> 6-very strongly disagree

RESULTS QUESTIONNAIRE - OPEN QUESTIONS BUSINESS USERS	
Q11	What is your job title?
Q12	What is your job function?
Q13	Based on your experience with LEAN, what is your opinion on its value as an enterprise architecture modelling tool?

Q14	What do you see as the strengths of LEAN?
Q15	Do you have any suggestions for improving LEAN?
Q16	What do you see as possible areas for the further development of LEAN?
Q17	Are there any other comments you would like to make?

RESULTS QUESTIONNAIRE - CLOSED QUESTIONS		
ENTERPRISE ARCHITECTS		
#	QUESTION	RESPONSE
1	Compared to other modelling languages I have used for enterprise modelling, LEAN is:	<input type="radio"/> 0-don't know <input type="radio"/> 1-extremely easy to learn <input type="radio"/> 2-very easy to learn <input type="radio"/> 3-easy to learn <input type="radio"/> 4-difficult to learn <input type="radio"/> 5-very difficult to learn <input type="radio"/> 6-extremely difficult to learn
2	LEAN is an effective language for modelling high-level (conceptual) information.	<input type="radio"/> 0-don't know <input type="radio"/> 1-very strongly agree <input type="radio"/> 2-strongly agree <input type="radio"/> 3-agree <input type="radio"/> 4-disagree <input type="radio"/> 5-strongly disagree <input type="radio"/> 6-very strongly disagree

3	LEAN is an effective language for modelling low-level (detailed) information.	<ul style="list-style-type: none"> o 0-don't know o 1-very strongly agree o 2-strongly agree o 3-agree o 4-disagree o 5-strongly disagree o 6-very strongly disagree
4	LEAN captures information across all technical domains of interest.	<ul style="list-style-type: none"> o 0-don't know o 1-very strongly agree o 2-strongly agree o 3-agree o 4-disagree o 5-strongly disagree o 6-very strongly disagree
5	LEAN captures information from all business areas of interest.	<ul style="list-style-type: none"> o 0-don't know o 1-very strongly agree o 2-strongly agree o 3-agree o 4-disagree o 5-strongly disagree o 6-very strongly disagree
6	LEAN leads architects to think more deeply about the structures and relationships that exist.	<ul style="list-style-type: none"> o 0-don't know o 1-very strongly agree o 2-strongly agree o 3-agree o 4-disagree o 5-strongly disagree o 6-very strongly disagree

7	LEAN models convey more meaning than the models I previously used to describe enterprise architectures.	<ul style="list-style-type: none"> o 0-don't know o 1-very strongly agree o 2-strongly agree o 3-agree o 4-disagree o 5-strongly disagree o 6-very strongly disagree
8	LEAN models convey meaning more precisely than the enterprise architecture models I previously used.	<ul style="list-style-type: none"> o 0-don't know o 1-very strongly agree o 2-strongly agree o 3-agree o 4-disagree o 5-strongly disagree o 6-very strongly disagree
9	I would use LEAN again for enterprise architecture modelling.	<ul style="list-style-type: none"> o 0-don't know o 1-very strongly agree o 2-strongly agree o 3-agree o 4-disagree o 5-strongly disagree o 6-very strongly disagree
10	I would recommend LEAN for use by enterprise architects.	<ul style="list-style-type: none"> o 0-don't know o 1-very strongly agree o 2-strongly agree o 3-agree o 4-disagree o 5-strongly disagree o 6-very strongly disagree

RESULTS QUESTIONNAIRRE - OPEN QUESTIONS

ENTERPRISE ARCHITECTS

Q11	What is your job title?
Q12	What is your job function?
Q18	What other languages have you used for enterprise modelling?
Q13	Based on your understanding of LEAN, what is your opinion on its value as an enterprise modelling tool?
Q14	What do you see as the strengths of LEAN?
Q15	Do you have any suggestions for improving LEAN?
Q16	What do you see as possible areas for the further development of LEAN?
Q17	Are there any other comments you would like to make?

13 APPENDIX D – USDOS ITA



The State Department web site below is a permanent electronic archive of information released prior to January 20, 2001. Please see www.state.gov for material released since President George W. Bush took office on that date. This site is not updated so external links may no longer function. [Contact us](#) with any questions about finding information.

NOTE: External links to other Internet sites should not be construed as an endorsement of the views contained therein.

United States Department of State Information Technology Architecture



Volume One ITA

April 16, 1999
Version 2.2

Table of Contents

1 INTRODUCTION

- 1.1 What is an Architecture?
- 1.2 Why Have an Architecture?
- 1.3 Benefits of the ITA
- 1.4 Architecture Management and Development Process

2 BASELINE ISSUES AND LIMITATIONS

3 ITA STRUCTURE

- 3.1 Architecture Layers
- 3.2 Technical Reference Model and Standards Profile
- 3.3 ITA Integration

4 TARGET ARCHITECTURE OVERVIEW -- STRATEGIC VALUE TO THE DEPARTMENT

- 4.1 ITA Support for Strategic Department IT Goals
- 4.2 Guiding Principles
- 4.3 Functional Model of Target ITA

5 ARCHITECTURAL LAYERS

- 5.1 Business Architecture Layer
- 5.2 Department Mission and Priorities
- 5.3 Business Drivers
- 5.4 Common Business and Information Flows
- 5.5 Technical Architectural Layers
 - 5.5.1 *Overview of Target Technical Architecture*
 - 5.5.2 *Technical Reference Model*
 - 5.5.3 *Information Architecture Layer*
 - 5.5.4 *Description*
 - 5.5.5 *Applications Architecture Layer*
 - 5.5.6 *Description*
 - 5.5.7 *Infrastructure Architecture Layer*
 - 5.5.8 *Description*
 - 5.5.9 *Hardware Platforms*
 - 5.5.10 *Infrastructure Services*

6 IMPLICATIONS OF AN IMPROVED ARCHITECTURE FOR USERS, EXECUTIVES, AND SYSTEM MANAGERS

- 6.1 Users' View
- 6.2 Executives' View
- 6.3 System Managers' View

7 NEXT STEPS

- 7.1 Review Process
- 7.2 Detailing of the Architectural Layers
- 7.3 Development of Key Segment Architectures

8 ACKNOWLEDGEMENTS

Annex A, Glossary A-

Annex B, Current Issue Paper Candidates B-

Annex C, Architecture Segments C-

Table of Figures

- Figure 1, Relationships between the ITA and Other Department Processes
- Figure 2, The Baseline -- Islands of Automation
- Figure 3, Modest Improvement in IT Integration
- Figure 4, ITA Structure
- Figure 5, Example of Relationships among ITA Segments and Layers
- Table 1, IT Goals Linked to Representative Architectural Features
- Table 2, Guiding Principles
- Figure 6, Functional View of Target Architecture
- Figure 7, Department of State Business Processes
- Figure 8, Business and Information Flows -- Transactional
- Figure 9, Business and Information Flows -- Collaborative
- Figure 10, Target Environment -- Emphasizing Commonality
- Figure 11, Overview of Target Technical Architecture
- Figure 12, Technical Reference Model
- Figure 13, Overview of the Information Architecture Layer
- Figure 14, Structure of the Information Architecture Layer
- Figure 15, Structure of the Applications Architecture Layer
- Figure 16, Conceptual View of the Infrastructure Architecture Layer
- Figure 17, Infrastructure Components

1. INTRODUCTION

The Department of State is pursuing the modernization of its Information Technology (IT) infrastructure to improve the technical support it provides to overseas posts and domestic mission and administrative operations while taking advantage of new information technologies.

Two key trends affect the future strategy of IT use at the Department: (1) the globalization of accurate, timely, and usable information, enabled by worldwide availability of modern commercial networks; and (2) the decentralization of computer resources, as reflected in the pervasiveness of the personal computer and networks that interconnect them. These two trends create a challenge for the Department in managing decentralized resources in a way that supports the global access needed to support the mission and business needs of end users. This challenge gives rise to an urgent need for an enterprise-wide Information Technology Architecture (ITA).

1. What is an Architecture?

An architecture is a guiding strategy or framework. As in a building project, the architecture represents the bridge between the customer's requirements and the technical design that will effectively satisfy those requirements. It is not a detailed blueprint or wiring diagram, understandable only to technicians, but rather more like the city planning codes, zoning laws, and high-level plans that constrain the design, and enable the objective to be realized. The Department's ITA provides guiding principles and standards to be applied when designing and implementing information services for Department users. The ITA also specifies the major components of the technology infrastructure to be built to support business requirements.

2. Why Have an Architecture?

An architecture is a prudent management tool that will help ensure that IT is responsive to Department business requirements. It will help the Department achieve technology goals and objectives cost-effectively by providing the basis for Department-wide coordination of IT activities, and a set of standards and common technical services that will foster interoperability and information sharing, while lowering total cost of ownership. In particular, the architecture will promote such benefits as broad access to information, efficient re-use of IT components and solutions, and effective global management of IT support. In addition, Federal law mandates an ITA for the above reasons in the Information Technology Management Reform Act of 1996 also known as the Clinger-Cohen Act.

The Department of State's ITA adapts the architectural model endorsed by the CIO Council as described in the document, "Federal

Enterprise Architecture Conceptual Framework," dated August 1998.

The ITA has also been influenced by examples cited by OMB and the Gartner Group. Gartner defines an Architecture as a ". . .framework and a set of principles, guidelines or rules to direct the process of acquiring, building, modifying, and interfacing IT resources throughout the Enterprise. These resources can include equipment, software, communications protocols, application development methodologies, database systems, modeling tools, IT governmental organizational structures and more." This definition is in keeping with the guidance presented in the 25 Oct 1996 OMB Raines memorandum "Funding Information Systems Investments."

3. **Benefits of the ITA**

A system and information management environment based on this ITA will have considerable benefits for the Department. It will promote the following:

- Universal access to corporate and global information sources to all authorized users
- Efficient and effective management and decision support
- A flexible platform to meet changing requirements
- Investment planning and cost-effectiveness in IT spending
- Security through proven commercial security solutions
- Consistency in how data is stored, shared, and appears in user applications
- Effective integration of new IT

1. **Architecture Management and Development Process**

Over the past few years, the Department has put a series of planning and management processes in place to govern IT investments. These efforts have been successful in putting the current modernization program on track, and have produced significant results, such as the worldwide deployment of the ALMA-based infrastructure.

The IRM Office of Architecture and Planning (IRM/APR/IAP/AE), working with senior management and Bureau representatives, will establish and document a set of interrelated processes for planning and managing development projects to ensure conformance with the ITA. The overall systems life-cycle portion of this process is illustrated by Figure 1, Relationships between the ITA and Other Department Processes, which includes the following key elements:

- Close integration with IT strategic and tactical planning
- Configuration Management (CM) of the ITA itself, so that changes are made in an orderly, well-reasoned manner
- Extensive ongoing Bureau involvement in planning and CM processes, through representation on Configuration Control Boards and Technical Review Advisory Boards convened to address specific architectural issues
- Cross-project coordination to minimize duplication of effort and promote reuse

- Project and system assessment to ensure conformance to the ITA

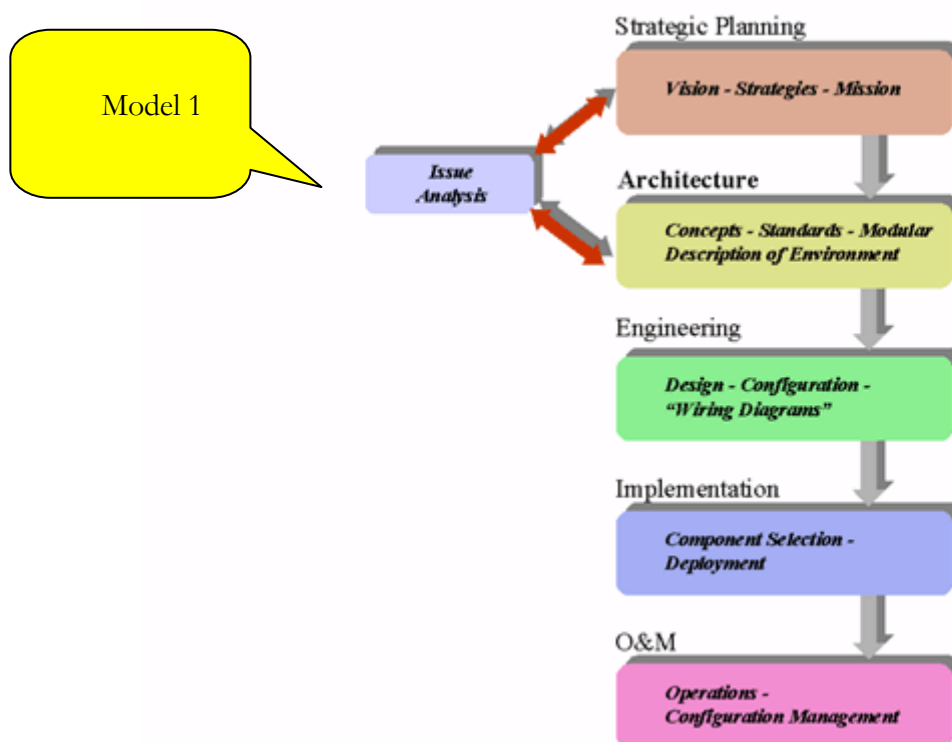


Figure 1, Relationships between the ITA and Other Department Processes

As shown in the figure, the ITA is tightly integrated with the Department's planning, engineering, design, and development processes. Strategic goals and plans specify what the Department intends to accomplish in IT and how those accomplishments will support business and mission operations. These plans also identify IT-related issues that must be analyzed and resolved in the course of developing and maintaining the ITA. Since the ITA is an "evolving description of an approach to achieving a desired mission," as defined by the GAO, it is necessary to guide the evolution of that description in a structured, predictable way. The exploration of topics that will result in updates to the architecture is maintained in a separate series of documents -- issue papers or "white papers" -- only a few of which may be active at any time. The topics that influence the architecture derive from the major system drivers -- mission or requirements changes that must be addressed and technology opportunities that may be exploited.

The following issues illustrate the types of architectural issues to be addressed:

- The nature and scope of the tools available to support applications development and the commonality of their usage
- The use of standard solutions for workflow, image management, and similar cross-cutting applications requirements

- The reengineering of messaging to meet all future requirements and exploit modern IT effectively
- The application and management of data warehouse technology
- The evolution of the standards for the products that are chosen to be the hardware, software, and networking components of the system
- The scope and operational approach to be applied to enterprise network management
- The technical and physical solutions for addressing security requirements
- The need for evolving customer service and user training with each IT enhancement

The issues to be addressed will result in decision support papers to be presented to appropriate levels of management for action. When a decision has been made, the ITA will be adjusted appropriately.

The ITA is the foundation on which IT solutions of the future will be designed, developed, and deployed within the Department. A key element in the Department's modernization program is establishing a modern, flexible, "open" platform or infrastructure to support all requirements. This will be done through platform engineering, which refers to the development of a common infrastructure for all applications in the Department. The platform is the stable, cross-project base of both infrastructure hardware and software provided to (and evolved for) all projects in the Department. Note that this contrasts with some commonly held definitions that consider "platform" to include only the hardware base.

Engineering leads into design, implementation, deployment, and operations. These activities are the responsibility of operational units in IRM (for infrastructure components) and the bureaus (for business applications). Guided by the Department's strategic plan, IRM plans, and ITA, these activities will promote a rational and integrated IT environment that responds to Department goals and priorities.

The processes for maintaining the ITA and for ensuring conformance will be documented in a separate IAP document that describes the roles of the Architecture and Planning Divisions, the capital planning boards, and other organizations. The general approach is for all projects to be reviewed for compliance with the ITA prior to submission to the IRM Program Board for approval and funding. Deviations from the ITA must be addressed by the project manager to the CIO. This review process will explore necessary changes to project plans as well as needed revisions to the ITA. The ITA will also be reviewed on a regular basis to ensure that it remains current with Department direction and priorities and with technology trends of industry and other agencies.

1. BASELINE ISSUES AND LIMITATIONS

Analysis of the Department's information technology baseline has identified several key issues that can be addressed by this ITA. These issues are discussed in this chapter.

IRM activities in the Department have historically been carried out on a decentralized basis and without the benefit of continuing centralized management attention. As a result, many development efforts have not been fully synchronized and the systems produced have not been fully interoperable. Figure 2, The Baseline -- Islands of Automation, presents a conceptual view of the past environment.

The current infrastructure, databases, and application systems have not been driven by an enterprise-wide architecture, and exhibit lack of commonality, interoperability, or portability, as one would expect. Such systems are described as "islands of automation" and "stovepiped" -- two metaphors that refer to their fragmentation and independence both in lack of commonality and in interoperability. The structure of the software that runs on the hardware platform is not guided by any perceivable enterprise-wide guidance.

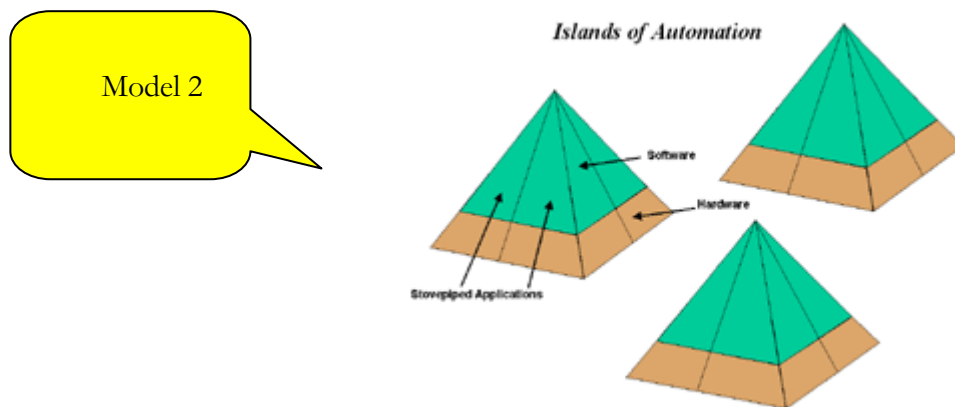


Figure 2, The Baseline -- Islands of Automation

Although the Department has made significant progress through the ALMA modernization efforts, the older environment remains plagued by components that are non-standard and not fully interoperable. As a result, end-users find it difficult to identify and locate information of interest, and collaborative processing is severely limited. Transaction processing systems use non-standard approaches and user interfaces, increasing the training and maintenance burdens for the Department. The following limitations characterize the current baseline environment:

- Lack of formal comprehensive IT standards
- Dependence on obsolete and non-standard equipment
- Obsolete software and cumbersome business processes
- Inadequate domestic and overseas communications circuits
- Non-standardized data
- Slow, unreliable, inefficient worldwide messaging, made up of multiple, non-standard E-mail and formal messaging systems held together by gateways and manual effort
- A workforce of system users and IRM professionals that is insufficiently trained in modern technology and operations

As noted above, the Department has begun to move purposefully into a "common component" architecture, with the adoption of ALMA. Figure 3, Modest Improvement in IT Integration, shows this modest level of commonality being achieved by the Department's current initiatives, which reduces the isolation of the former "islands of automation" by providing the benefits of common components, and the adoption of the software layering described later in the reference model. The earliest effects of platform engineering are starting to emerge for the Department as the layers above hardware are intentionally managed across all systems, and standard software building blocks, based on the standards profile, are adopted for common use.

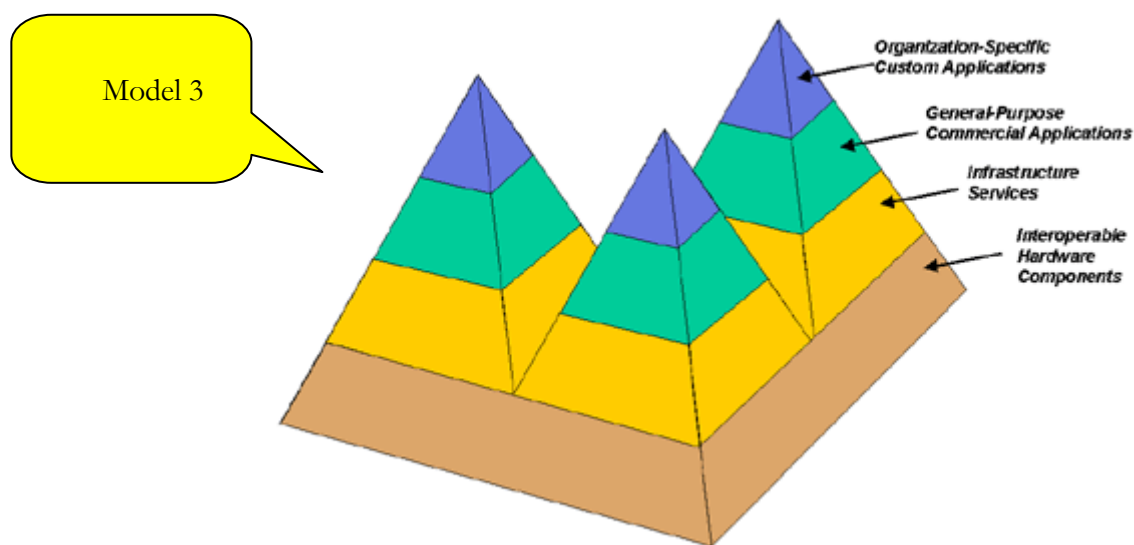


Figure 3, Modest Improvement in IT Integration

In the overall migration of ITA-based systems, additional commonality can be attained, at least to the level of common infrastructure services. This may be the highest level at which enterprise-wide commonality can or should be achieved, although common support applications will find use in several-to-many projects, but probably not all.

A major goal of the ITA and follow-on platform engineering is to increase the commonality across information systems. The target architecture presented in the next section provides a conceptual view of an environment with high-levels of commonality that can support broad information access and greatly reduce the fragmentation caused by islands of automation.

1. **ITA STRUCTURE**

The ITA is structured to provide a clear specification of a target environment that will meet Department of State goals and requirements. The architecture

describes, at a high-level, the IT environment that underlies the future vision of the Department's information systems.

Figure 4 shows the structure of the ITA.

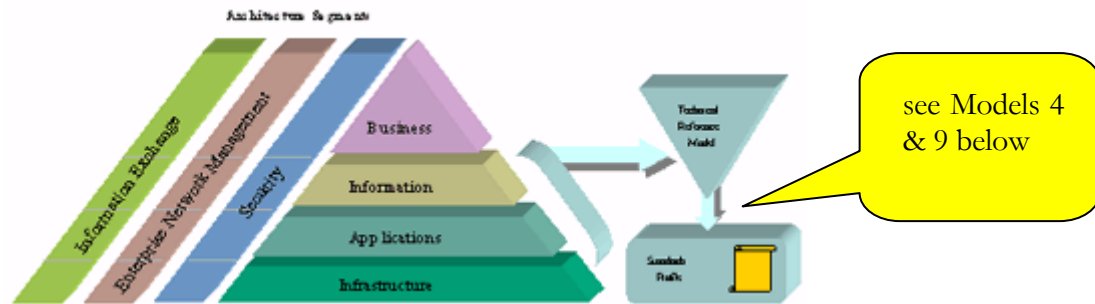


Figure 4, ITA Structure

1. Architecture Layers

The core of the ITA is the four architecture layers shown in Figure 4. These layers provide a framework for linking technical solutions to Department of State business requirements. The ITA supports both top-down and bottom-up planning and development of IT solutions. The top-down process links the business model directly to technical needs and solutions. Through bottom-up planning, the technical layers support identification of emerging technology trends and application of new technologies to mission needs.

Model 4

As shown, the ITA contains the following architecture layers:

- **Business** -- covering all aspects of the Department's business functions, including the overall flow of work and information, mission and management processes, and support functions used to meet user requirements at all levels.
- **Technical** -- consisting of the following three architectural layers
 - **Information** -- providing guidelines on the standardization, modeling, ownership, location, distribution, and access to corporate information.
 - **Applications** -- describing the environment in which applications will operate, the requirements and conventions to which applications must conform, and the services which application developers can expect from the infrastructure.
 - **Infrastructure** -- describing the network and its hardware and software platforms on which the IT infrastructure will be based.

The following provides an example of how the ITA conceptual model can be applied to a familiar, mission critical business function carried out by the Department every business day.

Model 5

A core **Business** of the State Department is the Non-Immigrant Visa Function, i.e. screening and providing proper documentation to aliens desiring temporary visitation to the United States. An objective of the Bureau of Consular Affairs is the speedy processing of legitimate travelers to facilitate travel to the U.S., but, as part of our Border Security function, to identify undesirables (including terrorists, drug traffickers, etc.) and refuse travel documentation to these undesirables.

Model 6

Visa processing is an **Information** driven process. The key data required is the applicant record that is collected from the applicant at the time of Visa application. Also, the Department maintains a database of people who are considered undesirables and who, barring special circumstances, should be denied entry into the United States. The database is populated from not only the Department's records, but also from information gathered from intelligence sources, law enforcement agencies, and other external sources. All Visa applicants must be checked against this database prior to being issued a Visa.

Model 7

CA has developed the Non-Immigrant Visa (NIV) software **Application** installed at all Visa issuing overseas posts. This software is a case tracking application that facilitates Visa processing by interfacing with the CLASS (Consular Lookout and Support System) database and other databases to help identify undesirables and, in so doing, enable Consular Officers to make informed decisions concerning Visa applications. Associated software applications provide long-term storage and retrieval of Visa records, and these applications also provide backup capabilities in the event that CLASS, which is accessed by long-distance telecommunication links, is unavailable.

Model 8

The NIV application installed at posts is an open system, standards-based application that complies with ALMA data processing, desktop, and communication standards. In fact, installation of the ALMA **Infrastructure** at an overseas post is a prerequisite to using the NIV application. NIV connects to the Washington-based CLASS application through OpenNet. Thus, OpenNet infrastructure is a requirement for effective Visa processing.

Model 9

The Visa issuance process, and many others like it, will continue to reap benefits through increased efficiency and sensible technology investments as specific ITA standards are articulated and implemented.

Architecture Segments -- The Department's ITA incorporates the concept of architecture segments, through which "hot topics" are addressed within the disciplined context and structure of the ITA. Not all architecture segments must be done at the same time or at the same level of detail. This allows "quicker returns" and early introduction of promising technologies. Figure 4 shows three initial segment architectures on the left side of the graphic. Additional segments will be identified as requirements become further refined.

- Security Segment -- Specifies security facilities and services to be provided by IRM and made available to all system planners and developers. The segment will describe the security infrastructure to be established (e.g., PKI, certificate management, firewalls, security service maintenance), as well as the specific security solutions to be deployed. It will address all security requirements to protect information, network, and system assets and provide information on

the integration of appropriate technologies to collapse the three separate enclaves. The security infrastructure will include a combination of solutions, each applicable to the specific security requirements of the data and circumstances. Thus, it will be more cost-effective and flexible than today's environment that relies on link/bulk encryption for most all situations.

- Enterprise Network Management Segment -- Specifies a Department of State Enterprise Network Management System (ENMS), a Department-wide resource implemented and maintained by the IRM Bureau to support both domestic and overseas operations. The system will provide various services to the bureaus including management reports, real-time device status information, help desk support through integrated databases and problem tracking resolution (PTR) systems, software distribution, and configuration and asset management tools and tracking. The Department expects the ENMS to improve network reliability, customer response times, and troubleshooting. Based on industry experience, the ENMS should also help contain total costs of ownership, (cost-avoidance), as the information technology (IT) environment in the field becomes increasingly robust and complex. The ENMS is being designed and implemented to support flexible arrangements with the customers -- the bureaus -- to enable bureaus to manage their own devices and applications if desired, while using the common infrastructure put in place by IRM. Conversely, IRM and the bureaus could establish Service Level Agreements (SLA) through which IRM would provide turnkey seat management services.
- Information Exchange Segment -- Describes a modern solution to replace the existing cable and email systems with an integrated document and information management and exchange solution that will support the Department in the 21st century. The solution will encompass the features of modern, business quality electronic mail, and will also support the diverse functions currently supported by the cable process (e.g., formal message and record traffic, policy dissemination, transaction processing). Although this segment will address the Department's current messaging systems, its scope is much broader. It includes future-oriented functions such as collaborative processing (groupware), document management, correspondence control, image management, archiving, and workflow-based transport of data and transactions (e.g., procurement or personnel transactions). Accordingly, the term "messaging" does not adequately describe the new, comprehensive environment envisioned.

1. **Technical Reference Model and Standards Profile**

The ITA also includes a *Technical Reference Model* (TRM) and *Standards Profile*. The TRM provides a linkage between the three technical layers and the set of standards contained in the Standards Profile. The TRM and active set of standards provide specific guidance to Bureau system managers and developers in the planning, acquisition, and implementation of IT solutions. In essence, they provide the most detailed and specific manifestation of the relationship between business requirements and technical standards.

The TRM provides the organization for the Standards Profile. Within each TRM category, the Department indicates the standards it has adopted. These may be true industry-wide standards, which transcend proprietary product boundaries, or they may be "product standards" -- solutions that the Department has chosen where no industry-wide standard exists. The latter are less desirable because they limit long-term flexibility and potential interoperability.

2. ITA Integration

The multiple components of the ITA are tightly interconnected, as illustrated in Figure 5, Example of Relationships among ITA Segments and Layers, which elaborates on the notion of segment architectures and illustrates areas in which segments overlap with the ITA layers. For example, the Security Segment focuses on delivering services such as encryption and user authentication, which can be used by systems across all architectural layers. While these services would be specified and developed within the Security Segment, their implementation is manifested in conjunction with specific infrastructure, applications, and information components.

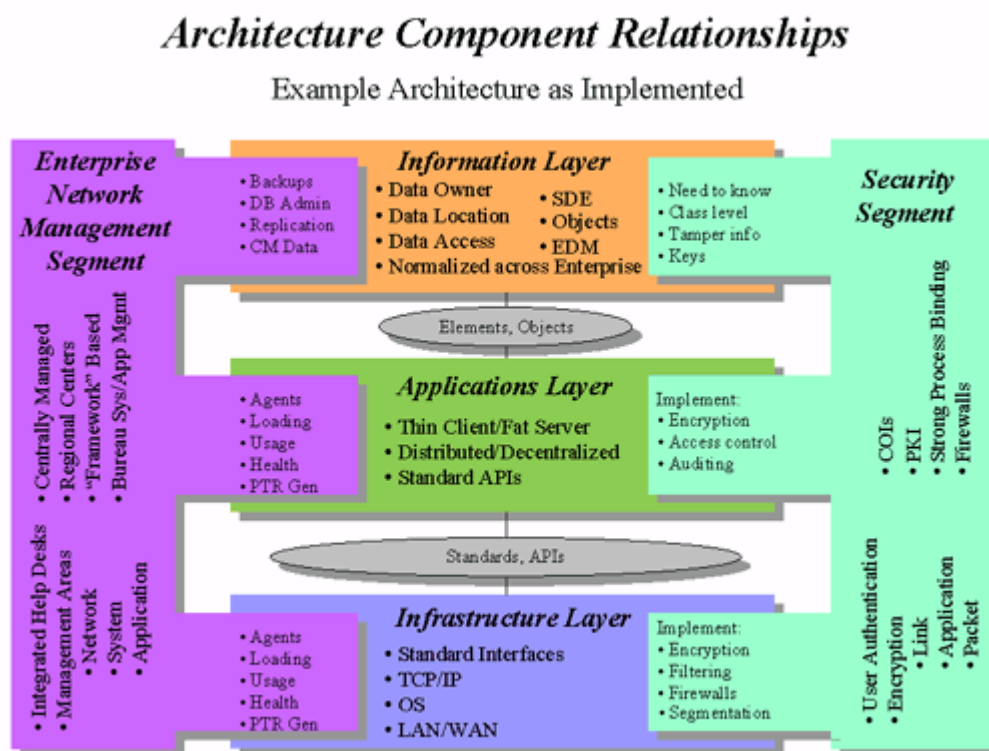


Figure 5, Example of Relationships among ITA Segments and Layers

1. TARGET ARCHITECTURE OVERVIEW -- STRATEGIC VALUE TO THE DEPARTMENT

1. ITA Support for Strategic Department IT Goals

The target Information Technology Architecture supports the Department's vision for IT in the new millennium. This vision calls for a robust, global, and secure information technology infrastructure that will support the growing and evolving demands of diplomacy in the 21st century. The core of the vision is a set of five goals:

- A secure, robust global network that links the entire international affairs community in the United States and around the world
- An expanded and readily accessible suite of systems and modern IT capabilities that support the substantive work of international affairs
- Modern integrated information exchange and document management ("messaging plus"), combining the best features of electronic mail, transaction processing, and collaborative document management and access
- Streamlined administrative applications that increase productivity and support our staff at all locations
- A trained and productive workforce, including IRM professionals to support the IT infrastructure, and well-trained end-users who can fully exploit the technology

The key elements of the ITA are derived directly from these goals. The business and technical architecture layers and architecture segments enable deployment of IT solutions that support these goals. Table 1 links the five goals and a representative set of architectural elements, showing how technical components contained in the ITA support achievement of the goals. The features shown are illustrative, rather than comprehensive. As the architecture evolves based on bureau feedback, the set of key features will expand and evolve as well. This table demonstrates how the ITA helps infrastructure engineers and designers identify technical solutions that address Department goals and priorities.

Table 1, IT Goals Linked to Representative Architectural Features

IT Goal	Applicable Architectural Features
Secure Global Network	<ul style="list-style-type: none"> • Robust, secure worldwide communications infrastructure • Commercial technology and protocols so network can evolve with requirements • VPN, PKI, satellite, Internet, and other technologies • Integration of all types of communications • Robust network security infrastructure
Suite of Foreign Policy Applications	<ul style="list-style-type: none"> • Consolidated information centers - "super servers" for access to corporate information • Knowledge management tools for highly intelligent user profiling • Web-enabled applications for ease of use

	<ul style="list-style-type: none"> • Searchable, accessible, secure data warehouse
Information Exchange -- "Messaging Plus"	<ul style="list-style-type: none"> • Business quality electronic exchange of messages and other information • Document and correspondence management products • Standards-based directory services • Security solutions for classified and unclassified information • "Thin client" enhances information security
Streamlined Operations	<ul style="list-style-type: none"> • Consolidation and standardization will yield efficiencies • Intelligent applications will simplify and streamline much administrative processing • Web-based processing will reduce complexity, especially at post • Infrastructure management will be automated and simplified • The IT infrastructure will be simplified through thin clients, centralized services, and standardization
Trained, Productive Workforce	<ul style="list-style-type: none"> • Centralization/regionalization will permit rational deployment of scarce personnel resources • Training will be enhanced through web-based technologies, distance learning, etc. • Simplified IT infrastructure management will reduce personnel burden • Global network may reduce need for on-site experts

1. Guiding Principles

The ITA will guide and support technology related decision-making throughout the Department. Bureaus will be guided by the standards, and guiding principles, and general approach in planning and deploying their IT infrastructures and specific applications and databases. Plus, corporate planners and senior management decision-makers, such as the IRM Program Board, will use components of the ITA to guide their deliberations.

Table 2, Guiding Principles, presents a set of guiding principles that underlie the ITA and subsequent systems acquisition and development.

Table 2, Guiding Principles

Architecture Layer	Guiding Principles
All Layers -- Overarching Principles	<ul style="list-style-type: none"> • Deploy and reuse modular components • Integrate security into all architectural elements, balancing accessibility and ease of use with protection of data • Strive for universal access to information • Limit complexity, especially at overseas posts. • Use off-the-shelf solutions where feasible • Focus on total cost of ownership and life cycle costs and benefits in planning and assessing IT solutions
Business Layer	<ul style="list-style-type: none"> • Allow mission needs and priorities to drive IT investments • Standardize business processes to gain efficiencies and ease the training burden
Information Layer	<ul style="list-style-type: none"> • Establish a corporate data model to enhance the value of the Department's information assets • Validate information once as close to its source as possible • Establish and use data warehouse(s) and corporate repositories • Minimize paper
Applications Layer	<ul style="list-style-type: none"> • Seek to reuse components from the Department's standard application libraries before developing or acquiring new systems • Employ Independent Verification and Validation (IV&V) for all applications before production deployment • Use web and similar technology to promote information access and standard user interface • Use the Department's standard suite of desktop and office automation products
Infrastructure Layer	<ul style="list-style-type: none"> • Establish a secure, integrated, reliable, high performance, global network • Provide all users with a common, standardized desktop • Migrate major assets such as servers into centralized facilities where they can be managed and secured cost-effectively • Provide a centrally managed Enterprise wide network infrastructure comprising LANs, MANs, and the WAN • Provide central facilities for help desk, network operations, security infrastructure, messaging, and other "core" services

1. Functional Model of Target ITA

Figure 6, Functional View of Target Architecture, is a high-level model of the target integrated architecture that provides a view of the functional capabilities the target architecture will support. As shown, the target will provide universal access to corporate information assets, and will also provide the security and technical infrastructure needed for internal and external connectivity. In short, the target positions the Department to meet its IT goals for the 21st century.

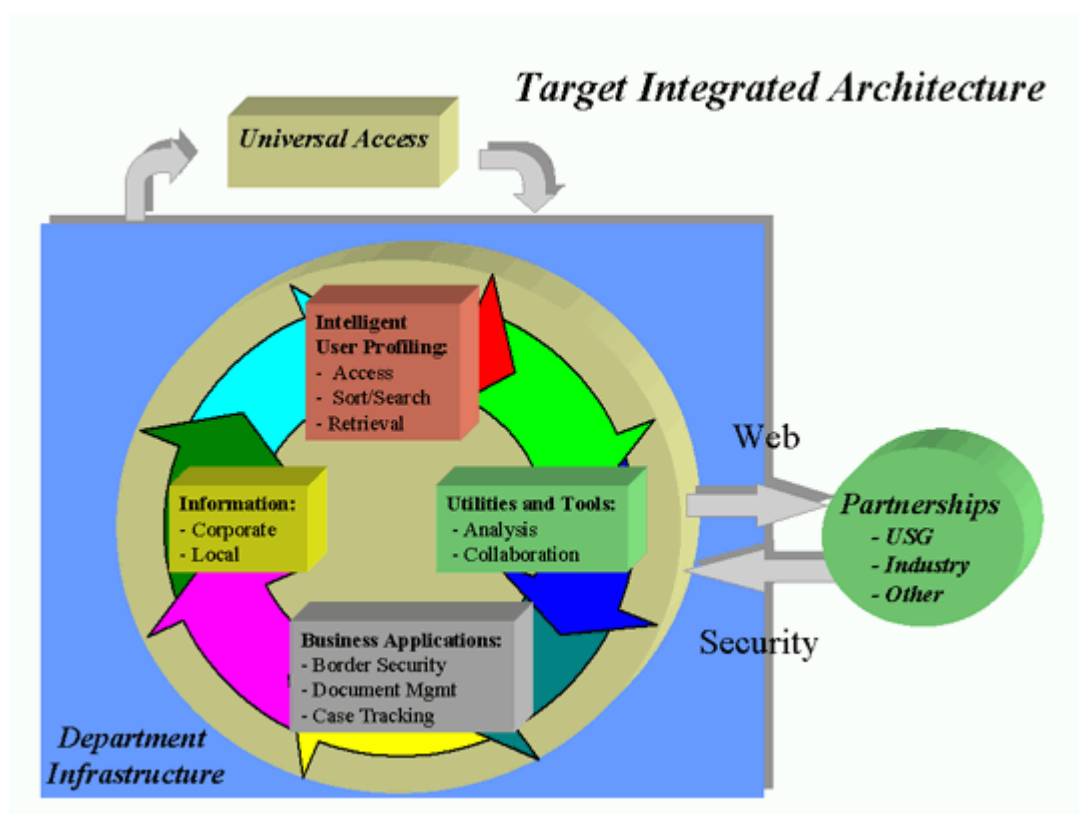


Figure 6, Functional View of Target Architecture

1. ARCHITECTURAL LAYERS

This section presents the two basic architectural divisions shown in the pyramid diagram presented as Figure 4, ITA Structure, above. The Business Architecture layer specifies the Department's major mission areas and business processes the Department performs. The business model leads directly to the establishment of a set of technical architecture layers, which consist of an Information layer, an Applications layer, and an Infrastructure layer.

Together these three layers specify a technical environment driven by and supportive of the Business Architecture Layer. The Business Architecture Layer drives the establishment of a corporate data model that is at the heart of the Information layer; a set of robust tools and application interfaces specified

Model 10

in the Applications layer; and an Infrastructure layer that enables the global connectivity and security so vital to State's effectiveness.

1. **Business Architecture Layer**

The Business Architecture Layer is a description of the mission-related activities and management functions performed by the Department of State, as depicted in Figure 7, below. It is based on the International Affairs Strategic Plan, the Department of State Strategic Plan, and Bureau and Mission Performance Plans, which identify United States national interests, strategic goals, and priorities for Department activities and investments. The Department's Strategic Plan provides the context for the future use of IT, which must be focused on furthering diplomatic readiness and the Department's mission and strategic goals.

2. **Department Mission and Priorities**

The Department of State's mission, derived from the President's constitutional authority, is to formulate and conduct the foreign relations of the United States. Within this broad context, the Department's focus has broadened considerably in the post-cold war era. While much activity continues to stress traditional nation-to-nation diplomacy, increasing attention is devoted to multi-lateral relations and global issues, such as international law enforcement, the environment, population, and terrorism.

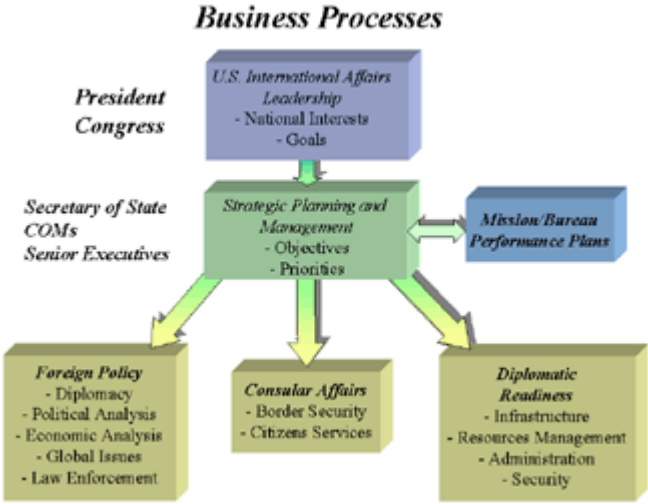


Figure 7, Department of State Business Processes

3. **Business Drivers**

The Department's mission and business environment is changing

dramatically as we approach the new millennium. As documented in several recent studies, the key challenge is to provide an environment that supports the new multi-faceted diplomacy in an electronic age -- *e-diplomacy*. To be effective in the next century, our diplomats will require access at their fingertips to a wealth of information and effective tools to manipulate that information and share it with others. To keep up with the issues of the day, they will require Internet-like networks and intelligent automated agents that can help them find and organize critical information. They will need the ability to collaborate with counterparts in other agencies, foreign governments, non-governmental organizations, and the public. Their work will become increasingly more dependent on information and IT networks.

4. **Common Business and Information Flows**

All of the Department's business processes can be represented by one of two information flow models: transactional, as illustrated in Figure 8, Business and Information Flows -- Transactional, and collaborative, as depicted in Figure 9, Business and Information Flows -- Collaborative, below. Transactional flows tend to be structured and Collaborative flows tend to be relatively unstructured and unscheduled, often resulting in free-form reports, presentations, issue papers, policy statements, and similar products.

The ITA's goal is to standardize common business processes and supporting technologies to the maximum extent possible. The ITA approach enables the Department to acquire and deploy common, general purpose "utilities" that can be used by any bureau or system manager to accomplish an identified business requirement.

This approach is analogous to industry-wide efforts to use and re-use web-enabled applications, or applets. General-purpose components can be plugged in or linked together to form complete system solutions with minimal programming and maintenance. The Department will accomplish this through applets and larger utilities -- for example, workflow, image and document management, search and retrieval -- which support business information flows and requirements.

Model 11

Business and Information Flows — Transactional



Figure 8, Business and Information Flows -- Transactional

Model 12

Business and Information Flows — Collaborative



Figure 9, Business and Information Flows -- Collaborative

5. Technical Architectural Layers

1. Overview of Target Technical Architecture

Model 13

The Technical Architecture describes the Department's information systems from a technical perspective. The systems correspond to and are driven by the requirements articulated in the Business Layer Architecture.

In contrast to the baseline and current transitional environment described in Section 2, Figure 2, The Baseline -- Islands of Automation (page *), and Figure 3, Modest Improvement in IT Integration (page *), the target technical architecture emphasizes commonality, a consistent use of standards, interoperability, and significant reduction in the prevalence of stovepipe solutions. Figure 10, Target Environment -- Emphasizing Commonality, illustrates the desired end state.

Model 3
(Duplicate)

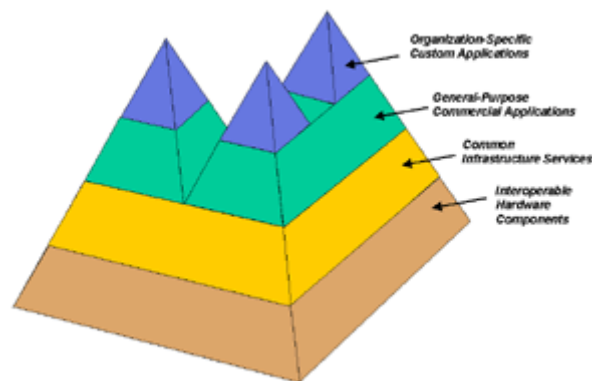


Figure 10, Target Environment -- Emphasizing Commonality

Figure 11, Overview of Target Technical Architecture, provides a high-level representation of information, applications, and infrastructure components supporting the target business architecture. This architecture is described in Section 5.1, Business Architecture Layer. The Department plans to establish a robust, reliable, and maintainable IT environment that provides the following key features:

Model 14

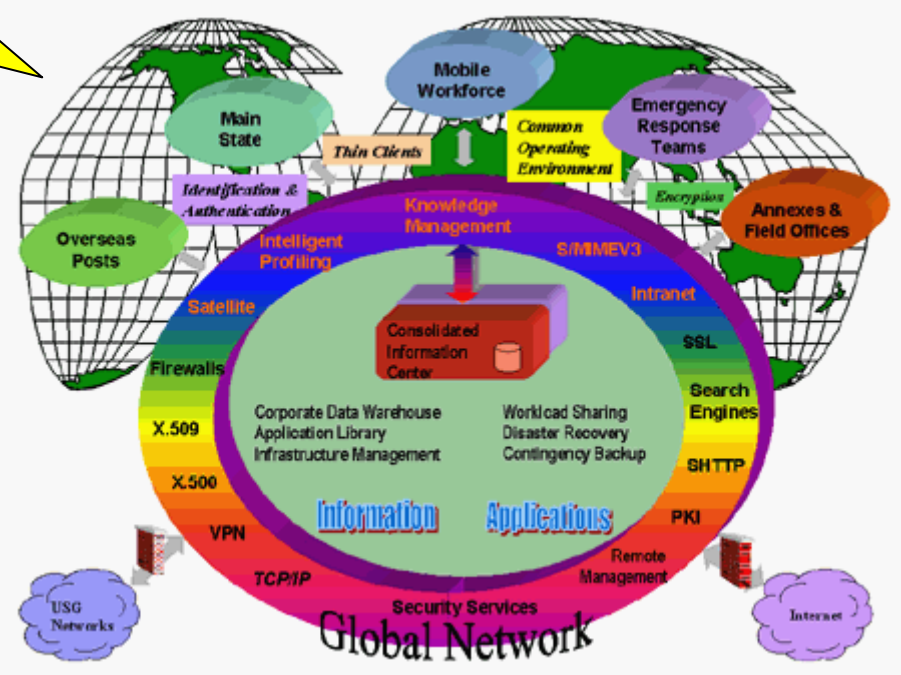


Figure 11, Overview of Target Technical Architecture

Model 15

"Secure Access to Information Assets from Around the World"

- **Integration and interoperability** -- while the current environment is plagued by duplicative systems and databases that do not work well together, the future environment will be characterized by consistent and highly integrated components such as Enterprise Resource Planning Systems. This includes integrated and shareable corporate databases, as well as the integration of voice, data, video, and other multi-media representations of information.
- **End-to-end connectivity** -- enabling staff to be able to communicate with each other and to access all enterprise information technology tools, data, and systems from their desktop. This entails a truly global network and global workgroup-processing environment.
- **Security** -- the Department's information resources will have strong built-in protections from internal and external threats while providing ease of access. Accordingly, the new environment will provide multiple levels of security to ensure the integrity of all IT components.
- **Manageability** -- the new modernized environment is being implemented in an era of shrinking budgets. Accordingly, the ITA specifies an environment that can be managed effectively with fewer resources. This will be accomplished through remote resource management using modern automated tools.

1. Technical Reference Model

The Technical Reference Model (TRM) serves as a bridge between the Technical Architectural Layers and the Standards

Profile. Figure 12, Technical Reference Model, is a depiction of the Department's TRM.

Model 16

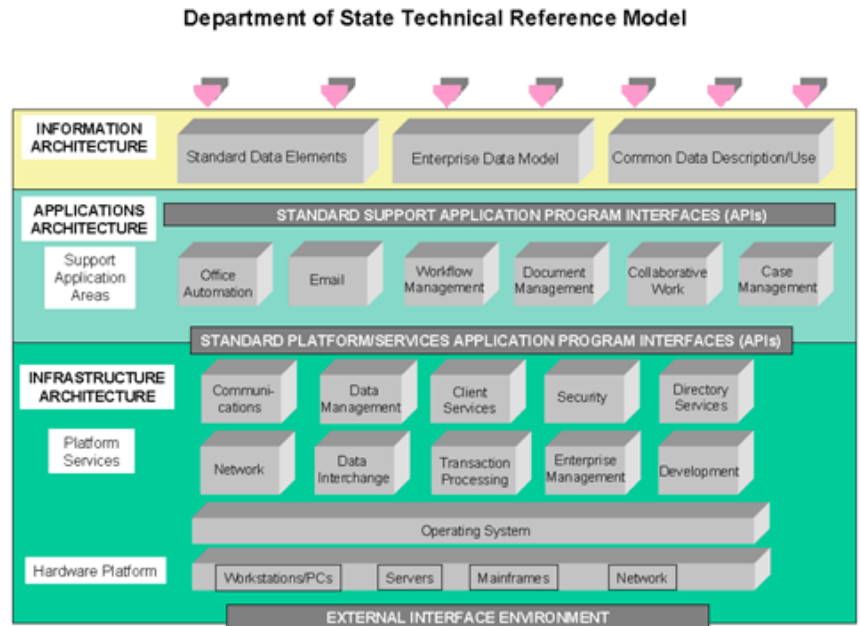


Figure 12, Technical Reference Model

The purpose of the TRM is to show the categories of entities in each technical architectural layer. That is, it shows the basic structure of the Information Layer and how it interfaces with the Business Architecture Layer, represented by the arrows on top of the figure, which identifies the major categories of information requirements. The Applications Layer portion of the TRM shows the top-level categories of applications to be used, both mission/administrative applications and support applications. Either mission or administrative applications may use the support applications area. The support applications may also supply common services to application software built on top of them.

The Infrastructure level of the Department's TRM shows both hardware and software components of the standardized system platforms. This includes all identified system services, broken out into ten categories that cover the full spectrum of platform services, including those not yet in the scope of the ITA, such as integrated voice and video. The hardware platforms themselves are shown as the lowest level, supporting the software levels above, typically through the operating system. These services are shown spanning the width of the diagram to

suggest that they are typically the direct interface to hardware functionality.

An additional feature of the TRM diagram is the "striping" between layers to indicate the standardization of interfaces across the enterprise. Thus, it is envisioned that a service level API will be established to invoke all system services in a consistent way. Managing this standardization and accepting or modifying existing software service interfaces are two of the ongoing architecture jobs over the system's life cycle.

The manner in which the TRM is used as a bridge to the Standards Profile is that each box in the TRM diagram represents a category subject to standardization. These are the niches into which related standards are grouped. While not all areas will have the same number of standards -- some, in fact, may not exist -- they identify areas of potential standardization.

Each level of the TRM can be accessed from whatever level above it that requires the services provided by that level. That is, this is not an "ISO-style" protocol diagram, where each layer may call only the layer immediately below it.

2. Information Architecture Layer

The Information Architecture Layer is constructed to satisfy the requirements outlined in the Business Architecture Layer. That is, the organization and hierarchy of data in the information architecture standards and descriptions are determined by the mission structure. The business requirements' data flows drive the allocation, replication, and other characteristics of the databases defined for the enterprise. The Information Layer is a framework that contains the individual data models developed for each mission area, and also defines the interrelationships between those models. Thus, it can be used to standardize data for sharing across the entire Department. This layer also contains Department business rules based on process analysis and on the interaction of processes and data. Focusing on both data and process enables Department organizations to reengineer and streamline their operations and better align them with mission objectives and priorities. Finally, the Information Layer is a key driver in defining system and infrastructure requirements in the application and infrastructure layers.

Figure 13, below, is an overview of the Information Architecture Layer, which focuses on supporting the following business requirements:

- Data organization and management to facilitate broad access
- Intelligent tools and systems for user and information profiling
- Data standardization, through the use of Standard Data Elements (SDE), and the Enterprise Data Model (EDM) to allow information exchange and interpretation

- Data warehouse for storage and retrieval of selected corporate information
- Sophisticated tools for knowledge management -- information search and retrieval
- Information security and integrity

One key element of this layer is the specification of a corporate or enterprise data model. The Department's decentralized environment increases the need for a corporate data model to promote information sharing and full exploitation of data as a critical resource. At the same time, the decentralization of State programs and systems imposes challenges for obtaining commitment to the development of an enterprise-wide data model. The IRM Data Administration Office has developed an initial EDM that is the foundation of the Data Administration program. This model is a continuous work in progress, evolving as Department programs and requirements change. An issue paper is contemplated to determine the best approach to the ongoing evolution and most effective application of the data model.

Each of the individual databases utilizes the Department's SDEs to ensure the ability to effectively share data across the enterprise. Each data steward is responsible for complying with the data standardization effort, to promote interoperability with other corporate databases and with the enterprise-wide data warehouse, which is created from the EDM.

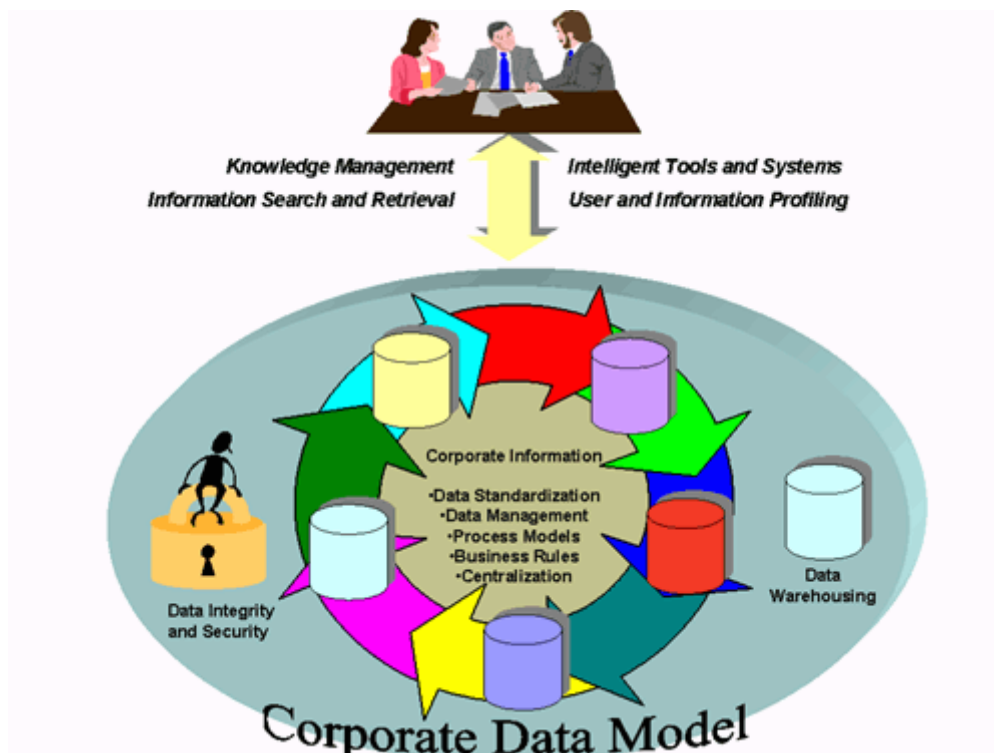


Figure 13, Overview of the Information Architecture Layer

The Data Warehouse must reflect the multiple views of data stewards who contribute to these stores, and deal with the potential that there are multiple sets of SDEs. The

location of the data warehouse may be central, or geographically distributed, for reasons of operational efficiency, redundancy, security, and safety. A key feature is that the warehouse(s) do not represent the superset of all corporate data, but rather a subset of critical data which may include data currently external to the Department that is of interest across the user community of individual databases.

1. Description

The top-level view of the Information Layer is a decomposition of the various subsets of data holdings in the Department as they relate to the separate mission areas defined in the Business model. Thus, Figure 14, Structure of the Information Architecture Layer, shown below, inherits its structure from the existing organizational and functional hierarchy. The Information Layer also contains additional components of the target architecture, notably cross-function data integration, multimedia (ensuring that the ITA addresses different formats for representing data, such as graphics, images, and video), and metadata (which contains information about the data and about the structure of specific data environments).

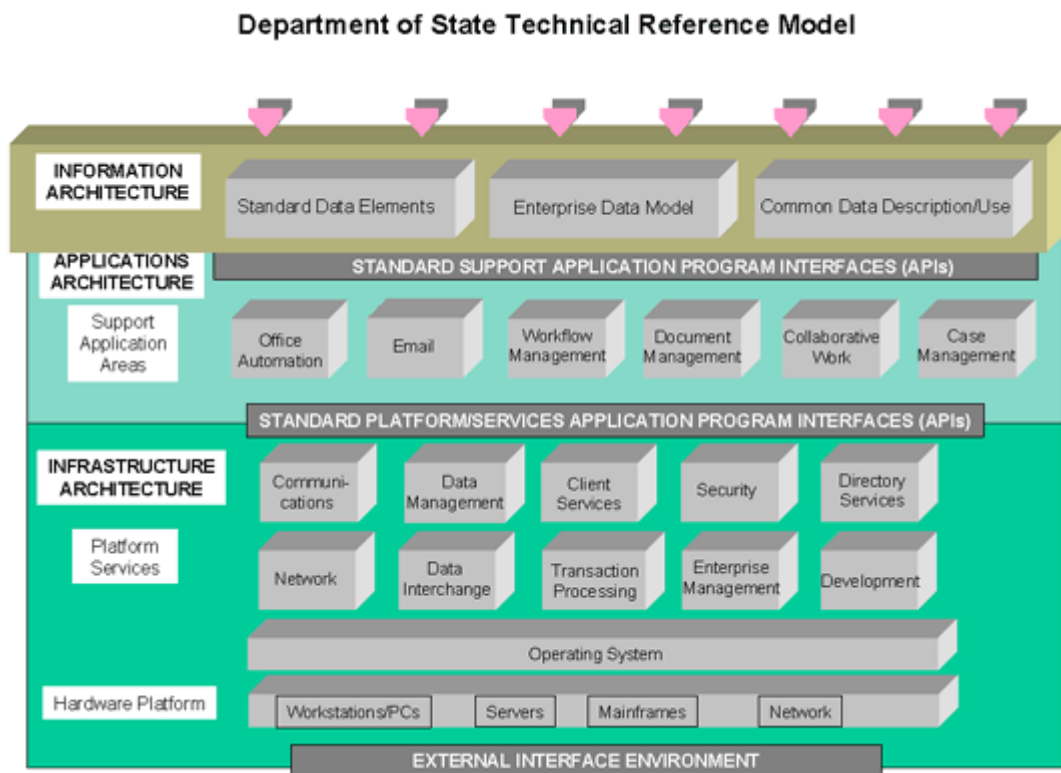


Figure 14, Structure of the Information Architecture Layer

The target information environment will provide enhanced support for Department requirements in user profiling and information access. Intelligent solutions will assist users in searching corporate data repositories and identifying information of interest. The system will be far more dynamic, flexible, and powerful than today's user profiling,

and will ensure that end-users have timely access to the information they need and are authorized to receive. The profiling capability will incorporate appropriate security safeguards to limit access to authorized users on a need to know basis.

The target environment will also support multiple techniques for viewing and presenting information. Thus, existing initiatives in geographic information and mapping would be extended to embrace new approaches and expanded data sets, as needed for Department programs and requirements.

When the Information Layer and its constituent data and process models are in place, individual database developers will apply these models to ensure interoperability with other databases and to guarantee the proper relationships with the data warehouse when it is instituted. All database designers and developers are to comply with the following guidance:

- Institute and enforce uniform data design management practices across organizational and project boundaries
- Use standard data structures, naming conventions, and SDEs
- Participate in development and maintenance of the Department of State EDM and use it in all application projects
- Establish retention periods for all data/information and purge or archive as soon as the period expires
- Designate a steward for all data to be responsible and accountable for: 1) data validity, 2) data source, 3) data definition, 4) data update frequency, 5) security and access rules, and 6) data protection levels
- Submit requests for new data elements to IRM/OPS/SIO/API (Data Administration) for validation of standardization, and subsequent submittal to the Data Administration Working Group (DAWG)
- Use shared source code libraries to promote good data management, there will be one and only one source code module for each individual data element

The Information Layer also contains process models that capture a clear understanding of the flow of work and information between and within organizations that combine with data models to accomplish the following:

- Track movement of data between mission/program areas
- Facilitate communication among data analyst, business user, and system developer
- Provide information on scope and boundaries, and identify integration points
- Clarify and consolidate user requirements
- Locate and record the data serving each business function

As noted above, this Layer also contains business rules, which are relatively permanent policies or constraints that govern and/or support business processes. Business rules are defined and recorded apart from the procedures and applications that use them.

1. Applications Architecture Layer

The Applications Architecture Layer describes the approach to be taken and services to be supported in acquiring, developing, and implementing application systems. The principles and strategies that underlie this layer reflect the business requirements and are intended to exploit the information architecture layer and modern IT technology effectively and efficiently. Features of this layer include standard utilities for the two modes of business process flows presented in the Business Architecture Layer -- transactional and collaborative. Some of the components of this layer are:

- Office automation
- Email
- Workflow management
- Document management
- Collaborative work
- Case management

1. Description

Figure 15, Structure of the Applications Architecture Layer, depicts the structure of the Applications Layer and suggests how it can be further refined as specific business and support applications are defined and developed. The overall structure involves two layers, or stripes. The "top" stripe is organized by business area, and the applications in each of those areas are constructed to satisfy specific functional requirements. These areas are not described in detail in this section -- they are included to reflect the link between the Applications Layer and the Information and Business Layers.

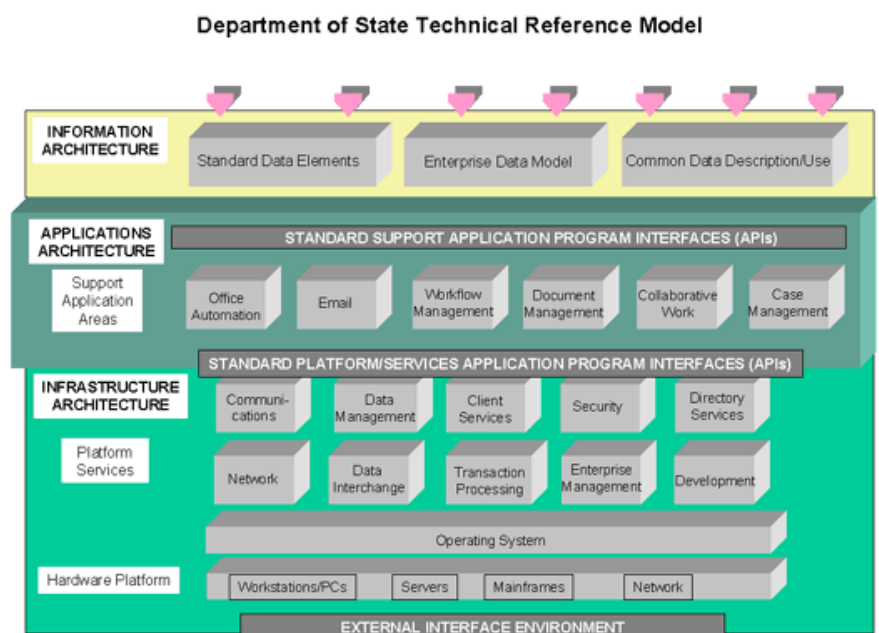


Figure 15, Structure of the Applications Architecture Layer

The lower stripe of the Applications Architecture Layer comprises the support application areas. These are sets of software products and services that are not structured along mission or functional lines. They may be commonly used in several, many, or all mission and administrative application areas. For instance, all users use email, and there is to be only one common email system in use across the Department. Others, such as case management might support only a few case-oriented user applications, such as in Consular Affairs, Legal, and Personnel. The same support application packages are available for use both in mission and administrative areas, and their use is strongly encouraged.

The Application Layer contains the standard utilities and tools needed to support the two basic types of business flows presented in the Business Architecture Layer. Bureau system developers will be able to use these utilities as building blocks to construct applications that meet specific functional requirements, resulting in a high-level of standardization. This should reduce development cost and training burdens, and facilitate interoperability and information exchange.

Support applications may be used directly by users, as with email and office automation. Or they may be building blocks for developing user applications. Workflow management and case management would fit in this category. Some support applications like document management might fall in either category, depending on how they are adopted. The applications layer makes these sets of support applications available to application development efforts, and designers must determine how best to use these assets. For example, message handling might be based on email and document management support applications, or it might find a more suitable mix of tools by using workflow management and collaborative work support applications.

A key feature of the set of support applications is that it is continually evolving. It is dependent on advances in technology and availability of workable tools that support Department business requirements. Indeed, Figure 15, above, shows a notional set of support applications. There may not be any

immediate need for certain of the technologies shown in the figure until future projects require them. But, they should be evaluated and selected in an architectural and enterprise-wide context.

The Applications Layer also includes two types of Application Program Interfaces (APIs), shown as the "stripes" in Figure 15. These APIs define standard interfaces between sets of software functions. The architectural concept behind an API is that standardizing the set of calls provides better enterprise-wide interoperability, supports portability across platforms, and protects the information systems from being captured by proprietary product suites. To the extent that industry standards are available they will be used. In other cases, the Department would define its own standards for the API.

The following paragraphs describe the currently defined Support Application areas.

Office Automation -- an integrated suite of desktop software available to end users (client software), providing basic non-mission-specific capabilities. Typically this includes network and file browsers, word processing, presentation-level graphics, spreadsheets, image viewers, personal (local) database management systems, media players, and utilities like calendars, schedulers, meeting planners, directories, calculators, file conversion and compression. The suite should be integrated and support linking between file types, such as seamless embedding of figures and spreadsheets in text, and creation of multimedia and hypermedia files.

Email -- the standardization of formats, protocols, and applications for sending, receiving, and manipulating electronic mail among individuals, organizations, and processes (programs). Email may be embedded in higher-level business-oriented applications, such as formal record message handling or administrative applications. It is also used directly as a utility user function, universally both within the Department and with external parties. And it may be used by other support applications as its communications mechanism of choice, for example, to move documents under a document management paradigm.

Workflow Management -- standards-based utilities that provide for creation, management, manipulation, and communication of work objects. Workflow management provides a high-level set of tools for dealing with sets of data and activities as they are described in systematic, transaction-oriented workflow

models. Workflow management may be useful for supply chain and other external party interactions, as well as structured internal functions like personnel and logistics.

Document Management -- the integrated collection of all functionality relating to creation, coordination, distribution, dissemination, storage, retrieval, version control, and disposition of documents; supporting a wide variety of transaction and document types. Rather than address different document types with separate systems, document management seeks to integrate data holdings, and provide consistent tools to monitor and manage the enterprise document structure. The full range of modern content and formatting options is available without regard to the historical accident of particular message or document formats or mechanisms.

Collaborative Work -- tools for interactivity among parties who need to share data or functions are contained in this category. They work on the holdings of the document or case management software, and they provide the underlying tools to manage the diversity of media that are managed by the document and workflow managers. This support application area addresses sharing, coordination, and presentation aspects of the shared work environment. The tools may be used as embedded features of other support applications, or they may be used as stand-alone programs, such as video teleconferencing support.

Case Management -- high-level management functionality focused on handling sets of related transactions against case folders. These would be applicable to personnel, legal, or consular affairs applications that must coordinate and consolidate sets of transactions and processing made in various places by multiple agents regarding an individual or situation. Case processing would also be useful for crisis, event, or topic-oriented situations where multiple parties take independent actions against a single set of related files.

Additional areas will be defined as required, and others may be eliminated when we reach a consensus that their functionality has been subsumed by other areas or is no longer of interest to the Department.

2. Infrastructure Architecture Layer

The Infrastructure Architecture Layer organizes and provides guidelines for the technical underpinnings of the enterprise set of information systems. It describes the services upon which the applications are built and the platforms that provide those

services. As with the other "layers" of the overall architecture, it interfaces with whatever layer or level that requires its services, not just with the layer immediately above it. For example, operators may directly interface with operating system services, and any user may directly invoke security services, as well as support applications and user applications using system services via the service application program interface (API).

Figure 16, Conceptual View of the Infrastructure Architecture Layer shows the conceptual topology of the Infrastructure Layer.

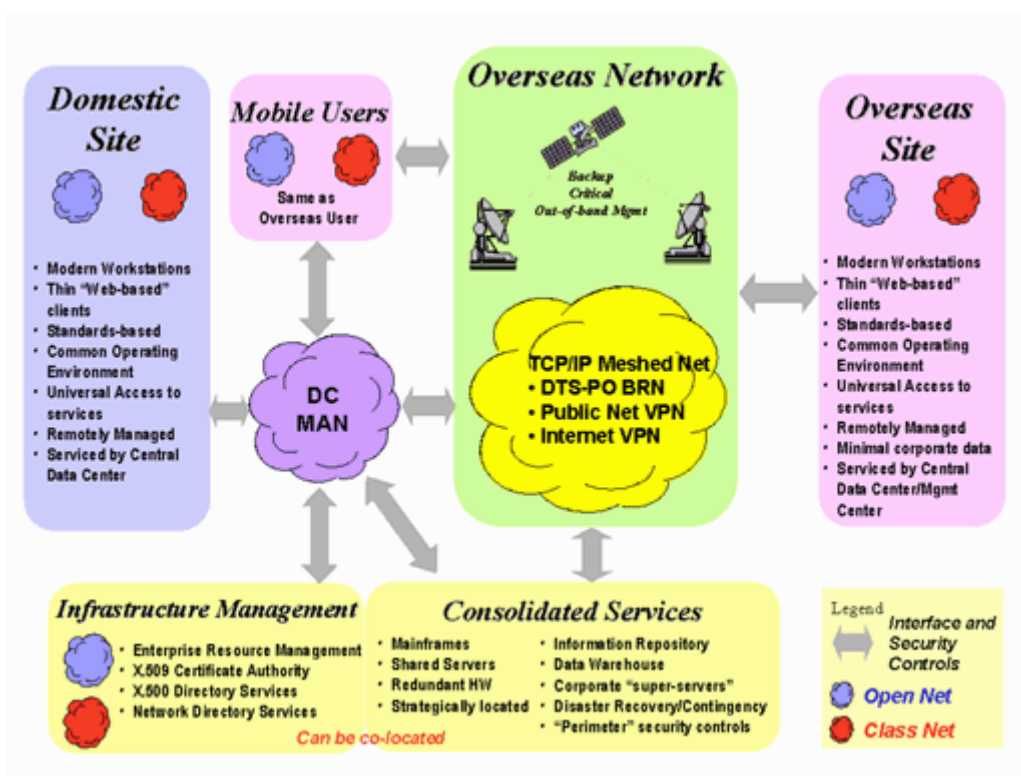


Figure 16, Conceptual View of the Infrastructure Architecture Layer

The Infrastructure provides the hardware and software platforms, network facilities, and associated services. The infrastructure reflects the target architecture and business requirements presented in preceding sections, including the following features:

- Centralized information centers to store and provide access to corporate information
- A secure, robust global network to support end-to-end connectivity
- An integrated solution for enterprise network management to ensure cost-effective support and maintenance
- Standards-based infrastructure services to promote interoperability and ease of maintenance
- Modern hardware platforms, including standard user desktops, thin client workstations, and computers for mobile computing

The infrastructure will have inherent attributes of reliability, scalability, flexibility, availability, manageability and maintainability. All these attributes presuppose commonality across the entire architecture from the user platforms to the Infrastructure required to support the Department mission.

1. Description

Figure 17, Infrastructure Architecture Components, depicts the components of the Infrastructure and their interrelationships. The basic organization involves two levels, a set of Platform Services (software functions) which are supported by the underlying Hardware Platforms. Platform components include all the hardware components of the system, from mainframes to PCs and everything in between, plus networking components.

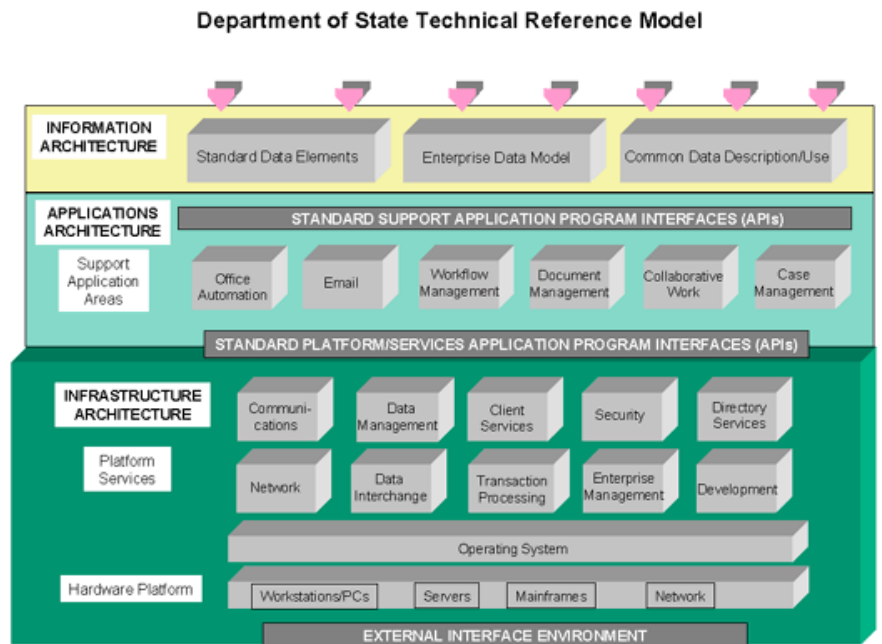


Figure 17, Infrastructure Architecture Components

The key interfaces are between the services and the applications that invoke them, and between the hardware platforms and the external world. The first is the Department-standard API that was discussed above in the context of the Applications Architecture Layer. As noted, it consists of standardized interfaces to the greatest extent possible, proprietary interfaces to the most limited extent possible, and additional interface modifications made by the Department to standardize the API where applicable. Part of the job of those who implement the Department's Infrastructure Architecture

is deciding on the level of API standardization appropriate for the required interoperability and portability needed.

As Figure 17 shows, the Infrastructure Layer contains two broad categories of components: hardware platforms, and platform services. The basic types of hardware platforms are derived from the Department's current environment. The number and organization of categories and components may change over time as Department requirements change and technologies emerge and change. The currently defined components are described below:

2. Hardware Platforms

Hardware platforms provide the physical component of the Infrastructure Layer. This includes all system components necessary to support the infrastructure services, and encompasses all physical interfaces between system components and external systems. These components include:

- Workstations and Personal Computers -- user workstations of any type, supporting client services; includes varying degrees of thin clients
- Terminals -- generally should be considered part of the user workstation component family, but may be called out separately to cover thin clients, Web TV, and network computers (NCs)
- Servers -- application, database, and other processing platforms, providing all the services and applications not allocated to the workstation components
- Mainframes -- although indistinguishable from servers in the strict architectural sense, the term is used to suggest the recentralization of server platforms onto large, centralized machines which only an enterprise data center can support
- Network components -- all network processors, interfaces, gateways, firewalls, hubs, routers, switches, and other protocol-bearing platforms used to support network services

1. Infrastructure Services

The Infrastructure Layer provides services either directly to end-users or to other IT components, such as applications and data repositories. The currently defined infrastructure services are:

Operating System -- Operating systems services provide the software environment and basic interfaces within all computing hardware platforms. They are the core services needed to operate and administer the platform and provide an interface between application software and the platform.

Communications -- This service group is a collection of platform services that are not supported by the Network Services group. For baseline and transition systems (prior to achieving a fully standards-based network system as defined in the target architecture), a number of other communications services must be provided, especially

to interface with external, non-networked systems. Besides legacy and non-standard communications interfaces, this group supports other non-digital data communications, such as voice and video, which are not fully integrated into the target network architecture

Data Management -- Central to information systems is data management, which is independent of the processes that create or use that data, maintained independently, and shared by an evolving set of processes. This service group encompasses the procedures, practices, methods, and software used to manage data, including data dictionary/directory, database management systems, and distributed data. The service group also encompasses any explicit data-oriented modeling tools not already provided by the Development Services group.

Client Services -- These services include user and applications interfaces that are often the most complex part of a system to develop and maintain. Within the past few years, significant advances have been made in user/application interface technology to enhance ease-of-use and to reduce the development effort required.

Security -- Security services support two common goals: Confidentiality and Integrity. Confidentiality provides the assurance that information will be held in confidence with access limited to appropriate persons. Integrity provides the confidence that information will not be accidentally or maliciously altered or destroyed. Security services provide functions to support both embedded functions (used by and within applications while they are running) and off-line, security analysis functions.

Directory Services -- This service group provides a common access point to user and other information for email, security, and other systems such as an electronic phone book. While formerly considered only a segment of network services, directory services are now recognized to have broader applicability across the application layer, and should be identified clearly for their role in security and enterprise management.

Network -- Network services provide connectivity and basic services to foster communications across workgroups and sites, supporting distributed data access and interoperability in a heterogeneous environment. Components of this category include data communications, electronic mail, transparent file access/transfer, remote network access, remote procedure call, and any other forms of inter-process communications.

Network Services is sub-divided as follows:

- Transport layer protocol specifications
- Network and link layer specifications
- Routing and control specifications
- General -- Related protocols and guidelines for IP networking
- Network Applications such as FTP and Telnet

Data Interchange -- Data interchange services provide specialized support for information exchange among applications on the same or different platforms. Components of this category include text data, spreadsheet data interchange, desktop

publishing interchange, graphic interchange, image compression/decompression, and calendar data. Extensions to full multimedia, and the metadata required to manage it and create hypermedia, are growing service areas within this group.

Transaction Processing -- Depending on the approach adopted by the Department, the transaction processing service group may or may not be an explicit component of the Infrastructure Layer. If an explicit transaction-oriented approach is adopted for those applications in the Department that are based on that sort of data interchange and processing model, then this service group would be used. A number of mature robust packages are available that could be adopted for Department-wide use, and significant savings could be achieved.

Enterprise Management -- Management services are integral to the operation of the Department's open systems environment. System management across the enterprise includes mechanisms to monitor and control the operation of individual applications, databases, systems, platforms, networks, and user interactions with these components. Management services enable users and systems to become more efficient in performing required work.

In addition to the embedded system management services, this category of platform services also encompasses Fault Management (including Help Desk), Configuration Management, Storage Management, and Capacity Management

Development -- Development services provide the structure to develop and maintain software that exhibits desired characteristics. This includes languages, tools, and methodologies, use of portable, scaleable, interoperable software. Development Services provide the infrastructure to develop and maintain software that exhibits the required characteristics.

In addition to software development and the support environment for code development, programming services provides the following support tools:

- Test environment; Test tools; Test case generation, execution, monitoring, and reporting
- Model development, simulation tools; Scenario generation/test case analysis; Integration tools

1. **IMPLICATIONS OF AN IMPROVED ARCHITECTURE FOR USERS, EXECUTIVES, AND SYSTEM MANAGERS**

While the ITA is primarily a technical document that guides IRM activities, it will have a profound effect on everyone in the Department of State. The following paragraphs discuss the ITA from the perspective or "view" of three different Department roles: end-users, executives, and system managers.

1. **Users' View**

By standardizing the Business Architecture Layer, users will find the Department of State's future IT environment more integrated and

flexible than what is in place today. Through standardization, business processes at State will mirror many of the functions employees can now do at home via the Internet. A more open environment will encourage greater information exchange, thus improving the quality and timeliness of information that is available to State users. In general, the user will see an IT environment that is less fragmented, more robust, more standardized, and more effective than what their workplace provide them currently.

The Information Architecture Layer will ensure that users have broad access to corporate information repositories, and that needed information will be represented in standard, easy-to-understand ways. Information needed to perform your job will be consistent, reliable, and organized as a mirror of the structure of your business processes and associated data flows. Web-based search engines will provide transparent access to the vast storehouse of information available within the Department and elsewhere on the World Wide Web.

Users will see the structure of the Applications Architecture Layer reflected in their own bureau applications (either mission or administrative functions for each user's needs). This would include direct access to support applications like office automation and email, and indirect use of other support applications, such as workflow management and collaborative work tools. Applications will be highly standardized throughout the Department -- when moving from Bureau to Bureau, the learning curve will be much less steep than it is today.

The Infrastructure Architecture Layer ensures proper security to authenticate users, some data management functions, and network directory services. The routine interactions that users have with the services provided on their workstations (PCs) will shape their view of the Department's IT Infrastructure. All other services will be (and should be) transparent to users.

2. **Executives' View**

Due to standardization within the Business Architecture Layer, Department managers will experience an environment that is more efficient, performance-driven, and competitive. While resource limitations will continue to put pressure on managers to deliver more with less, streamlined processes and reduced bureaucracy will increase each manager's control and potential effectiveness. To accomplish these ideals, managers will have access to powerful tools for resource planning and management.

Managers will expect the Information Architecture Layer to enable cost-effective use and access to corporate information. They will be able to obtain timely and accurate management information, as well as substantive data related to foreign policy research and analysis.

Through improvements in the Applications Architecture Layer, managers are concerned primarily with consistency, functional coverage, and training issues. Managers will be stakeholders in defining future functionality (more with upper levels than with support applications), but also will be able to influence decisions about application efficiency, interoperability, and change management. This architecture layer offers the potential to enhance the value of information systems, while reducing total costs.

Managers will use the Infrastructure Architecture Layer as the "whatever it takes to get the job done" aspect of the system. That is, managers will not care about how the needed functionality is accomplished, but only that, in the aggregate, the services are provided that support their users. This layer will ensure efficiency of operations, usability, and have an impact the selection of software and training requirements.

3. System Managers' View

Adoption of a standard Business Architecture Layer will allow system managers and developers to become solution integrators and consultants. As the Department embraces commercial technology and off-the-shelf solutions, we will do little custom software development. We will also integrate the Department's information assets by designing, developing, and maintaining one or more corporate data warehouses. IT professionals and users will need to develop the skills to support and utilize new approaches to accessing and leveraging information.

The Information Architecture Layer will provide system developers and IT project managers a basis for database design as well as applications and infrastructure development. The information layer establishes most of the service needs that must be provided. System managers will be required to coordinate database planning and design activities with the IRM/OPS/SIO Data Administration, to ensure standards compliance and conformance with the EDM, data interoperability, and warehouse standards.

The Applications Architecture Layer affects system managers most directly, since they are responsible for implementation and system operations. System managers will use standard utilities and other components and services available in this layer. As the Department moves toward standardization in application development and components re-use, system managers are expected to support these goals.

The Infrastructure Architecture Layer provides the basis for all platform engineering and development efforts of system managers and technicians for

the required operational characteristics of the system, including its functionality, performance, availability, usability, manageability, security, interoperability, and flexibility. System managers are expected to ensure that their specific solutions conform to the standards and approaches specified in the Infrastructure Architecture.

2. NEXT STEPS

This document is the first version of the Department's ITA. During the next six months IRM/APR/IAP/AE plans to transform this document via several iterations into the Department's official ITA. It plans to effect this transformation through three concurrent processes:

1. Extensive review of each version of the ITA document by IRM, Technical Review Advisory Board, bureaus, and IV&V reviewers, followed by modifications to the document based on comments and suggestions received.
2. Further detailing of the four architectural layers (Business, Information, Applications, Infrastructure), beginning with input from USIA and ACDA, but extending to whatever architectural details bureaus wish to have addressed.
3. Development of key segment architectures, such as Security, Enterprise Network Management, and Information Exchange that cut across the architectural layers and are considered important enough to the Department's IT activities to warrant special treatment.

The following paragraphs describe generally how these three processes will proceed and set forth target months for completing major elements of the ITA.

1. Review Process

As a general rule, reviews of the ITA document will proceed sequentially. This will begin with an internal review with the IRM bureau, and continuing through the Technical Review Advisory Board, bureaus, and IV&V. Depending on reviewer needs, IAP/AE will provide versions of the ITA in hard copy and electronic forms, make presentations, and arrange for group and individual discussions. Additionally, IRM/APR/IAP/AE will collect information and suggestions via questionnaires, interviews, and the Intranet.

Each new set of reviewers will see whatever improvements have been made to the document by the previous set of reviewers. In addition, whatever version of the ITA document being reviewed will include:

- Any detail that has been added to the four architectural layers (through concurrent Process #2)
- Any segment architectures that have been drafted (through concurrent Process #3) up to that point

In other words, over the next six months the ITA will be an extremely dynamic document, but reviewers will always have the most up-to-date and complete version of that document for their review.

We anticipate that bureaus will have several opportunities to review the ITA document and offer suggestions for its improvement:

1. Detailing of the Architectural Layers

IRM/APR/IAP/AE will add substantial detail to the descriptions of the four architectural layers that are set forth in this version of the document. The ITA will also be available on the IRM/APR/IAP Web site. This version of the ITA does not yet fully address the reorganization issues of ACDA and USIA, which will be addressed in the Business Architecture, nor does it necessarily include all issues of interest and importance to DoS bureaus. IAP/AE is holding discussions with bureau representatives to inform them of future enhancements of the ITA and to complete these four architectural layers

Target months for completing draft versions of the architectural layers are as follows:

Business Layer May

Information Layer July

Applications Layer August

Infrastructure Layer October

2. Development of Key Segment Architectures

IAP/AE plans to develop two segment architectures -- Security and Enterprise Network Management -in time for inclusion in the official ITA document to be published in October. Other segment architectures -- e.g. Information Exchange -- will be developed and added to the ITA document at a later time.

Target months for completing draft versions of the segment architectures are as follows:

Security May

Enterprise Network Management August

After the ITA is published its maintenance will be an ongoing process. As technology develops and provides improved methods for conducting business, the Department's ITA and associated standards will inevitably change. It is important that all elements of

the Department contribute to the development and on-going maintenance of the ITA, thereby assuring that the ITA and standards remain relevant to their needs in managing and using technology. Since this document represents the first step in shaping the ITA, it is particularly important that all bureaus participate fully in reviewing this document and suggesting improvements to it. For further information please contact the Architecture & Engineering Division Chief Greg Linden, IRM/APR/IAP/AE, at (202)776-8987 or email LindenGS2@state.gov.

Annex A, Glossary

ALMA	A Logical Modernization Approach -- Department model for automated information systems and telecommunications modernization based on use of Information Technology (IT) standards and commercial products.
API	Application Program Interface
Bandwidth	Term used to identify, or "measure" the capacity of a telecommunications circuit or local area network (LAN).
BRN	Black Router Network. A DTS-PO IP-based network in pilot phase -- The "Intranet" of the Foreign Affairs Community.
CM	Configuration Management
COE	Common Operating Environment. A term used to refer to a specific configuration for platforms such that all users utilize the same configuration thereby lowering management and troubleshooting effort and costs.
COI	Communities of Interest. In the context of this document, this term refers to a set of information, and the users, to which a group of users needs access. This concept is an extension of "need to know."
<i>e-Diplomacy</i>	The new multi-faceted diplomacy in an electronic age. To be effective in the next century, our diplomats will require access at their fingertips to a wealth of information and effective tools to manipulate that information and share it with others. To keep up with the issues of the day, they will require Internet-like networks and intelligent automated agents that can help them find and organize critical information. They will need the ability to collaborate with counterparts in other agencies, foreign governments, non-governmental organizations, and the public. Their work will become increasingly dependent on information and IT networks.
Encryption	The use of electronic coding techniques to protect information from disclosure to unauthorized readers, to prevent undetected modification of the information, and to support reader to writer identification and authentication.
Firewall	Any telecommunications or network device used to regulate/control the flow of information packets between networks. The firewall, or firewalls, implement an IT security policy by screening packets to verify they comply with policy, do not contain malicious code, and are not otherwise attempting to intrude on the protected network

side or disrupt its operations.

FTP	File Transfer Protocol
Intranet	An internal IP network. Open Net and Class Net are the "Intranets" of the Department.
IP	Internet Protocol - the basic standard established for data exchange over the worldwide Internet and widely adopted by organizations operating private networks, such as the Department's Open Net and ALMA-based LANs.
LAN	Local Area Network. A small network that serves a group of users. Typically confined to a single facility. Most LANs in the Department are built using Ethernet (10BaseT) hubs.
MAN	Metropolitan Area Network. A regional network that connects building LANs and backbones together and typically serves as a collection point to interconnect with a WAN.
OS	Operating System
OSI	Open Systems Interconnect
PKI	Public Key Infrastructure. The term used to refer the system required to supply and manage certificates for public key encryption and digital signature used by clients and servers
Platform	In the context of this document, the platform is the stable, cross-project base of both hardware and infrastructure software provided to (and evolved for) all projects in the Department. Note that this contrasts with some commonly held definitions that consider "platform" to include only the hardware base.
Protocol	A defined structure, content, and flow for communications between computers and other networked devices.
PTR	Problem Tracking Resolution. The business of tracking the process and information by which a help desk or other operational entity troubleshoots and resolves a user or infrastructure problem.
S/MIME	Secure/Multipurpose Internet Mail Extension. Provides a consistent way to exchange secure MIME data. Based on the popular Internet MIME standard, S/MIME provides cryptographic security services for electronic messaging applications: authentication, message integrity and non-repudiation of origin (using digital signatures) and privacy and data security (using encryption). S/MIME is used by traditional mail user agents to secure the text and attachments. However, S/MIME is not restricted to mail; it can be used with any transport protocol that transports MIME data, such as HTTP. As such, S/MIME takes advantage of the object-based features of MIME and allows secure messages to be exchanged in mixed-transport

systems. Further, S/MIME can be used in automated message transfer agents that use cryptographic security services that do not require any human intervention, such as the signing of software-generated documents and the encryption of FAX messages sent over the Internet.

SDE	Standard Data Element
SHTTP	Secure Hypertext Transfer Protocol. A means of securely transmitting HTTP formatted information.
SLA	Service Level Agreement - a definition of the type, quality, and quantity of network services agreed to by the provider and the customer.
SSL	Secure Socket Layer. A means of securely transmitting Web pages.
Standard	Agreement on the rules, procedures, and content of AIS and telecommunications exchanges to include open standards, industry standards, and de facto standards
TCP/IP	Transmission Control Protocol/Internet Protocol
Thin Client	In the context of this document, "thin client" refers to minimizing the amount of processing logic and data manipulation on a client to the maximum extent possible. The "thinnest" client is nothing more than a terminal.
VPN	Virtual Private Network - a capability to 'split' a physical network or circuit path into two or more sub-paths that use various protocols to define the circuit path and protect the data being transported.
WAN	Wide Area Network. This refers to a collection of circuits that interconnect a widely dispersed set of facilities, and other networks such as a MAN.
X.500	A CCITT protocol, X.500 is a family of standards and uses a distributed approach to realize a global directory service. Information of an organization is maintained in one or more so-called directory system agendas (DSAs). The X.500 directory supports a variety of services including security (certificates), e-mail (addressing), and "white pages" (name and phone number).
X.509	One of the X.500 standards that defines a security certificate to provide a vehicle for associating users with their encryption keys. All of the user's "public" information is stored in a X.509 certificate for use when exchanging information securely with that user. Other information such as to whom does the user belong, what authority issued the keys, when do the keys expire, what levels of information classification is this user allowed to access, and how can the certificate be validated is also included.

[End of Document]

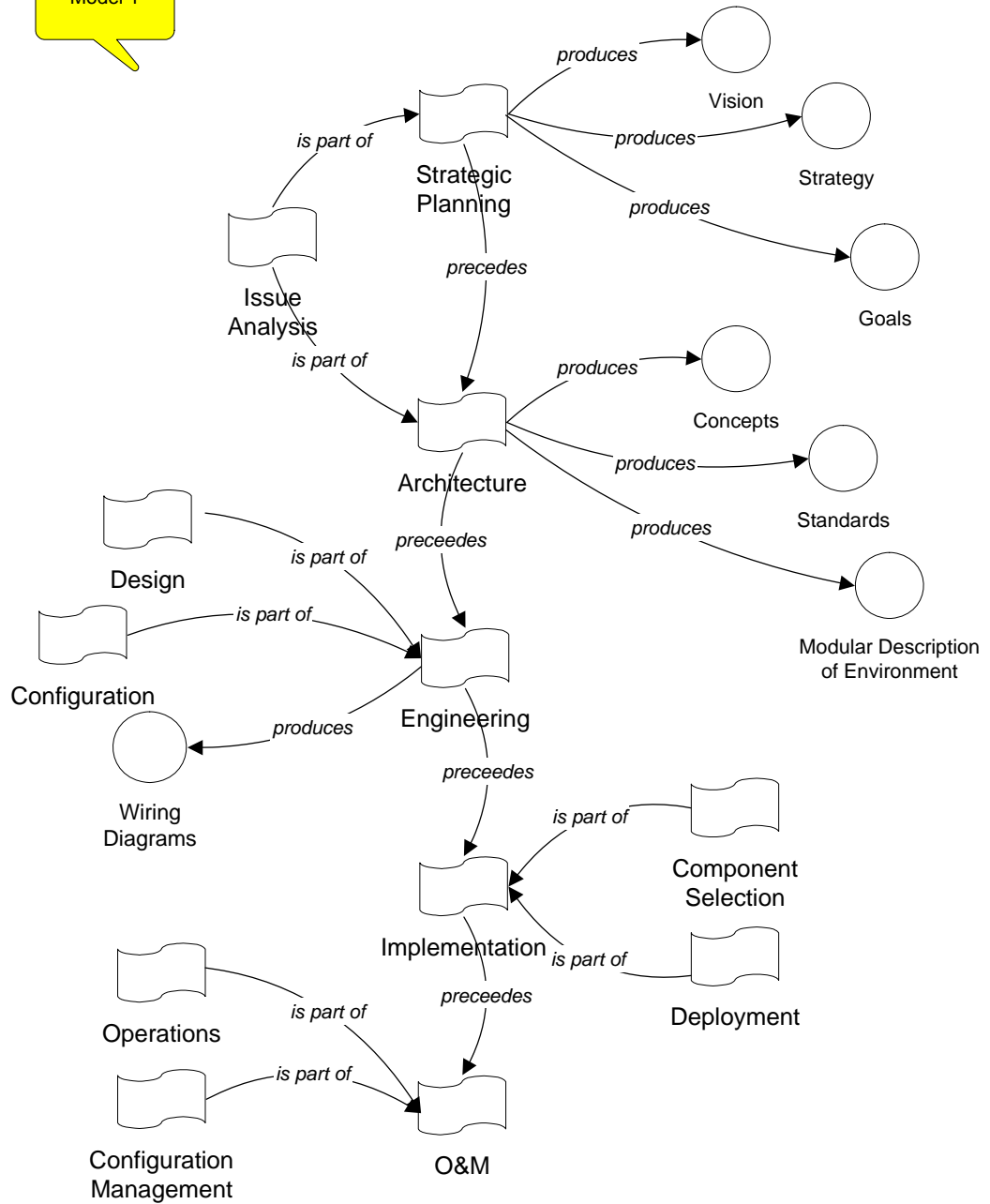
14 APPENDIX E – USDoS ITA LEAN MODELS

The following models have been translated into LEAN from models or textual descriptions in the USDoS ITA.

Following each model are some observations that arose from development of the model.

Model 1

Relationships between the ITA and Other Department Processes

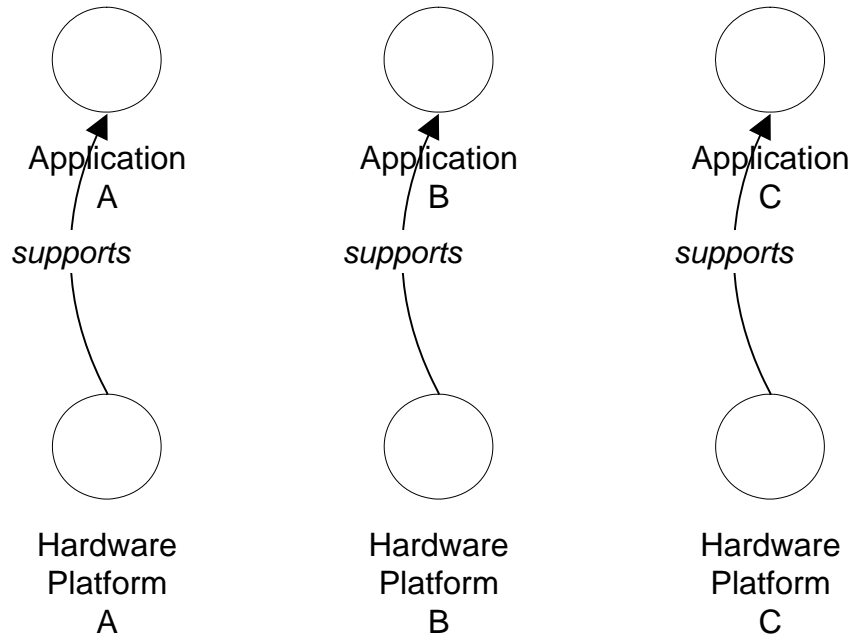


Observations:

- While this model shows that “Engineering produces Wiring Diagrams”, in actual fact, the Engineering department is likely to both use, and produce these diagrams. The original model provides no information on the relationship between Engineering and Wiring Diagrams other to suggest that there is some sort of relationship. In fact, the original model does not provide any information about the nature of any of the implied relationships.

Model 2

The Baseline -- Islands of Automation

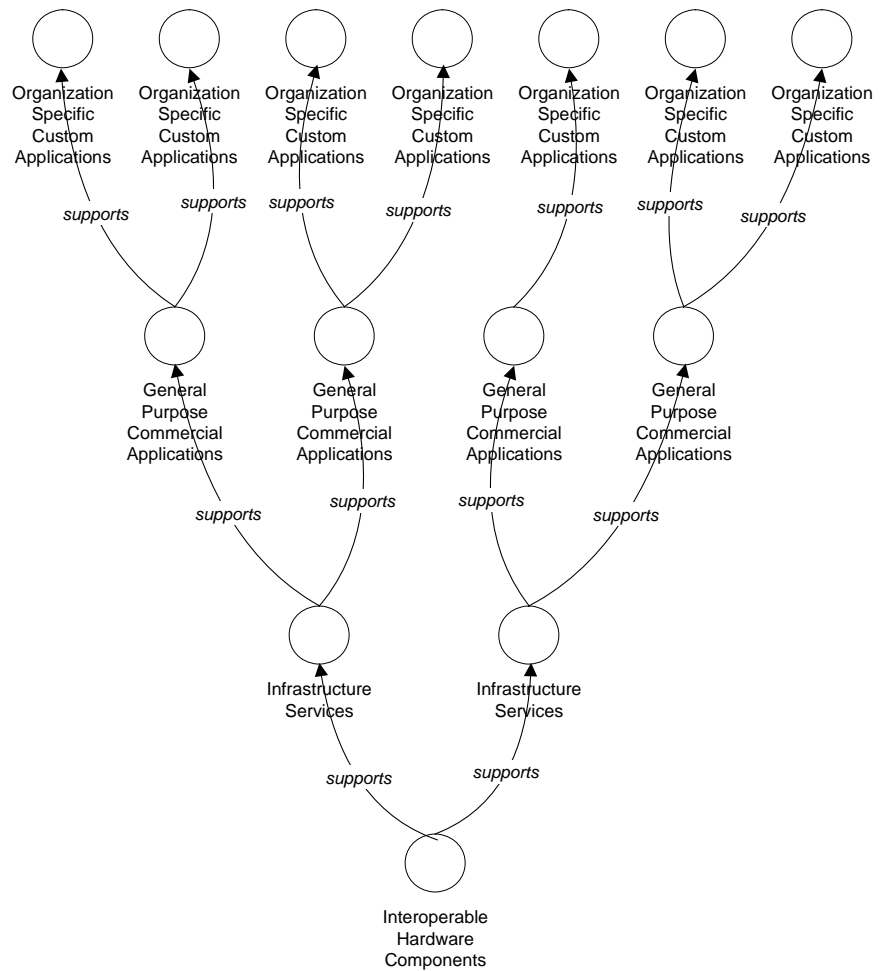


Observations:

- This model is designed to convey the metaphors of systems as “islands of automation” and “stovepiped”.
- While LEAN was not designed to convey these types of ‘soft’ concepts, it appears to convey these metaphors just as effectively as the original model.

Model 3

Modest Improvement in IT Integration

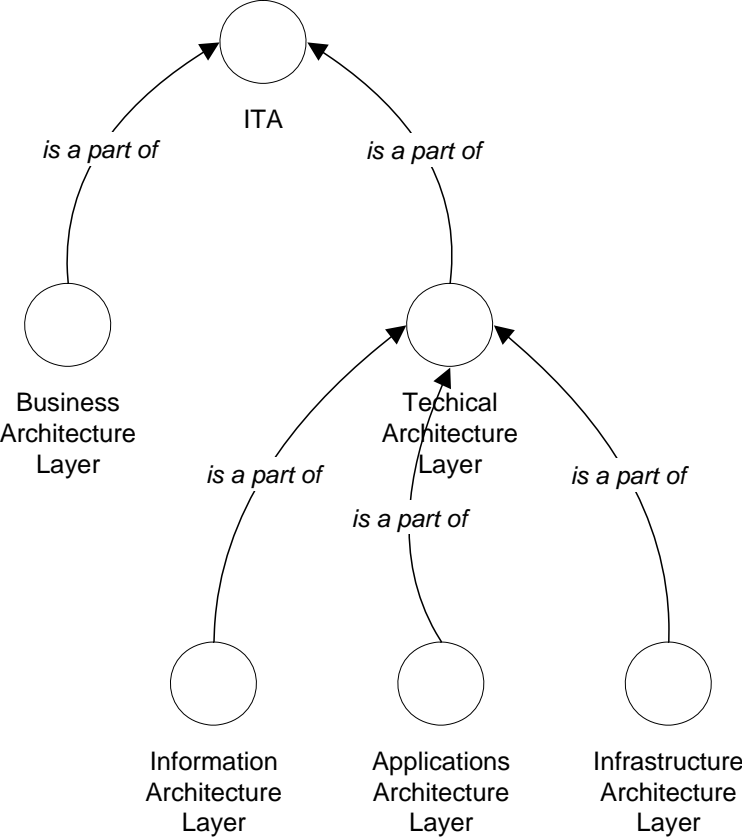


Observations:

- Some liberties have been taken with the production of this model: it is actually semantically different from the original. In fact, the two uppermost layers of the original model do not, I believe, accurately convey what the authors intended. The original model shows that multiple “Infrastructure Services” can run on one set of “Interoperable Hardware Components”. It seems that this same concept is meant to be applied to the two highest layers, but instead of showing *multiple* components being supported by each lower component, the higher components are shown as being ‘*smaller*’. The new LEAN model appears to convey the tree-like hierarchy of these layers more effectively.
- The danger with this LEAN model is that it could be taken more literally than is intended. That is, the number of systems represented is not meant to be literal, but merely to represent a general concept of decomposition.

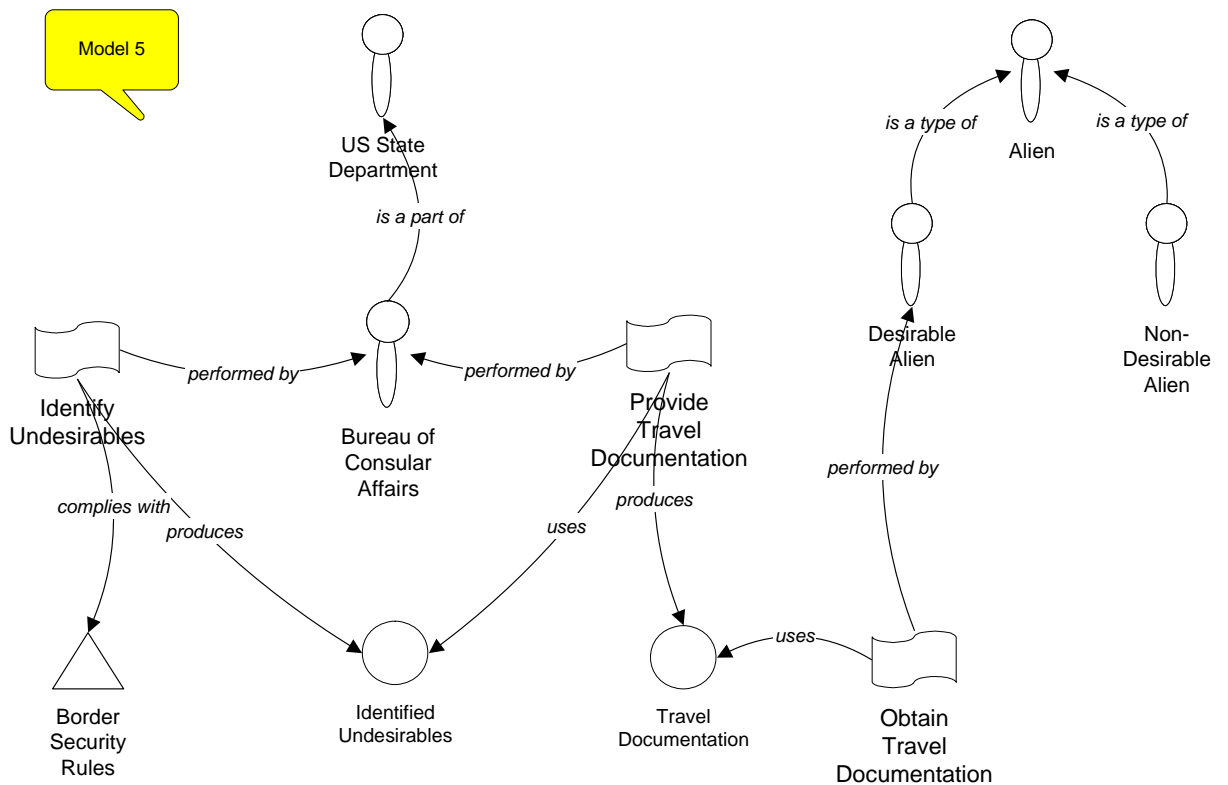
Model 4

Architecture Layers



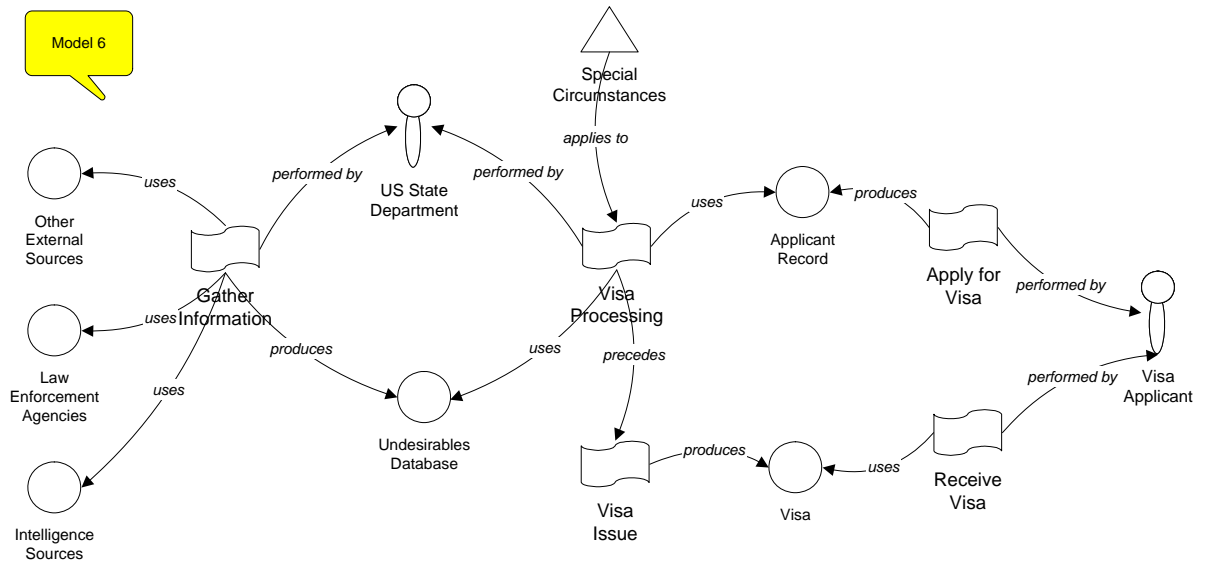
Observations:

- It is very easy to create a LEAN hierarchy that graphically illustrates the concept conveyed only textually in the USDoS ITA.
- The creation of these types of hierarchies is a common task in EA modelling. They are often represented using ‘block’ diagrams.



Observations:

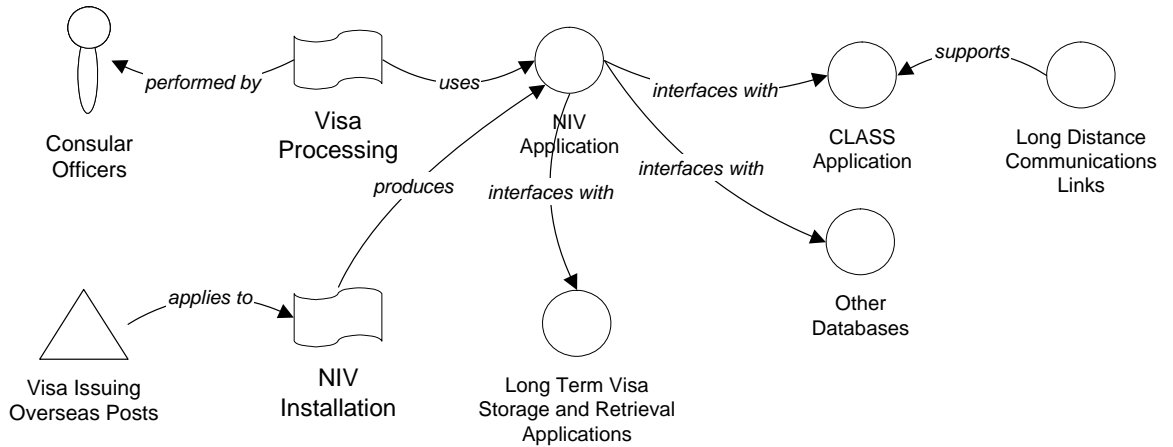
- This model shows how LEAN can be used to represent typical business scenarios. Such a scenario is typical of the type of information that would typically need to be modelled during EA planning and design phases.
- There is no one *correct* way to model this business scenario. The scenario can be interpreted in various ways, various inferences made, and emphasis given to different phrasing. For example, in the model above, no direct link has been made between the Agents “Bureau of Consular Affairs” and “Aliens”. Instead, the interaction is via the Resources that are used or produced, i.e. the data that identifies undesirable aliens and the production of travel documentation for desirable aliens. However, it could have been drawn differently with more direct links between these Agents. In a real-world EA endeavour, the modeller would draw the models, and decide which information to abstract away, in order to highlight the semantics that are most important to the business owners, while also providing a level of detail that supports the alignment of future, lower level models.



Observations:

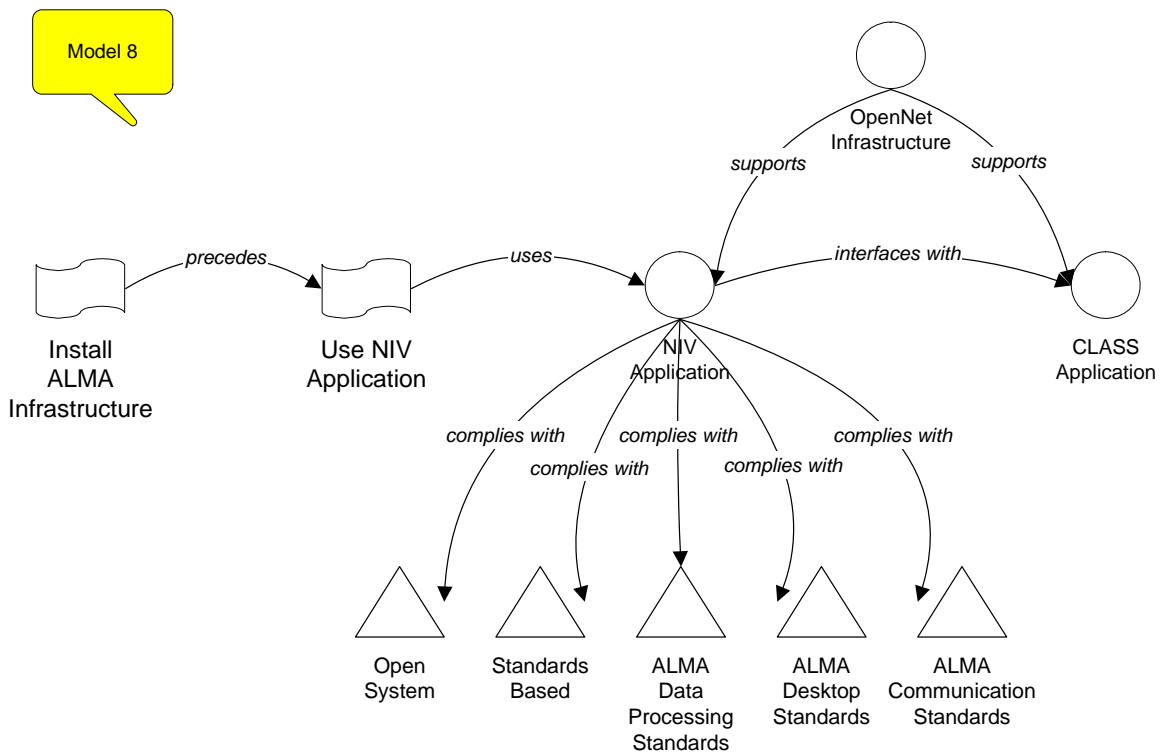
- There is a temptation here to create direct links between the “Visa Applicant” and “Applicant Record” and “Visa”. That is, to indicate that the applicant creates an application and then receives a Visa (or is rejected). The Generic Relationship Set does not provide a direct link between an Agent and a Resource. Instead, an Action has to be created as an intermediary. This is in keeping with the societal metaphor that LEAN is based upon. It is only through some Action, that Agents make use of Resources. The identification of these Actions makes explicit a feature of the system that may have been lost otherwise.

Model 7



Observations:

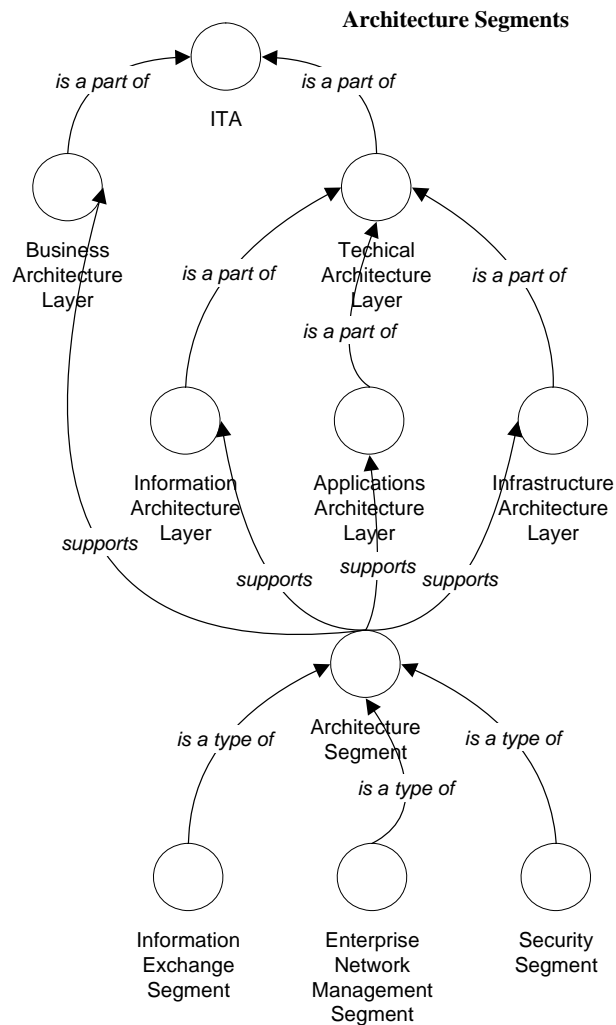
- This business scenario states that the NIV application is installed at all Visa issuing overseas posts. This was signified in the LEAN model by creating an Action “NIV Installation” with an associated Rule “Visa Issuing Overseas Posts”. This seemed to be the most effective way to illustrate the scenario using the Generic Relationship Set and, although slightly contrived, does seem to convey the semantics of the scenario without undue complexity.



Observations:

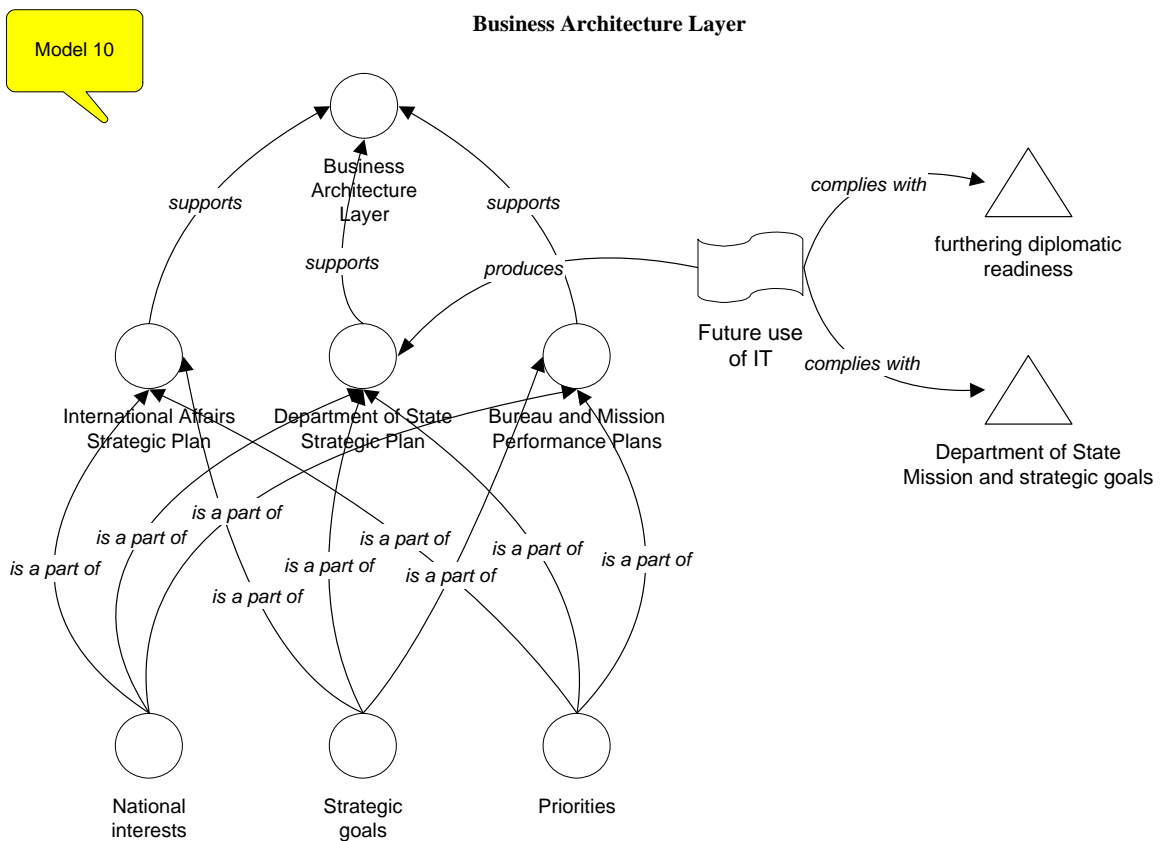
- The condition that “installation of the ALMA Infrastructure at an overseas post is a prerequisite to using the NIV application” has been shown through the temporal connection between two Actions, “Install ALMA Infrastructure” and “Use NIV Application”. This condition could alternatively have been represented using rules connected to the NIV Application Resource instead of using processes. Yet another alternative is to use a “Supports” relationship such as shown connecting the “OpenNet Infrastructure” Resource to the two resources it supports.

Model 9



Observations:

- The architecture segments that are referred to in this business scenario are shown in Figure 4 of the ITA specification, but since the modelling constructs used within the ITA document are not defined, it is difficult to ascertain how these segments relate to the architecture layers. This is typical of the problems that arise when using informal 'block' diagrams for EA modelling.
- Based on the textual descriptions of the architecture segments, combined with the preceding descriptions of the architecture layers, we can reasonably assume that the segments as represented in Figure 4 are meant to be interpreted as foundation technologies upon which all of the other architecture layers are built. LEAN can model these relationships by representing the architecture segments as resources.

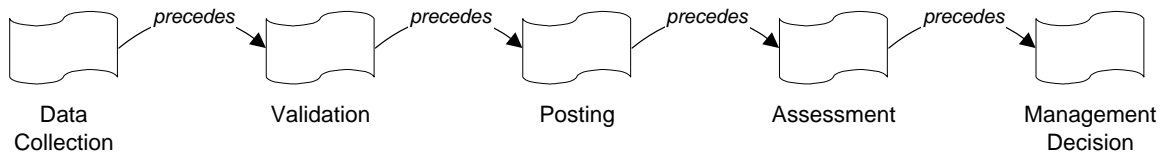


Observations:

- Although this paragraph in the USDoS ITA refers to Figure 7 in that document, there appears to be little connection between the description provided in text and the model. This LEAN models represents the Business Architecture Layer as described in the text.
- This model has become a little complicated due to the many arcs needed to represent many-to-many relationships. These could have been avoided by a new, artificial construct such as “part of plan”, but this approach would make it unclear which parts applied to which plans.

Model 11

Business and Information Flows - Transactional

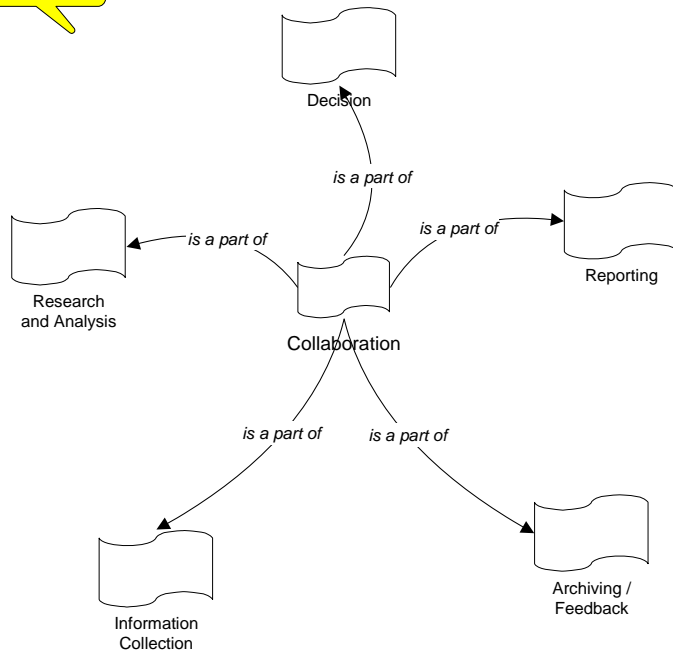


Observations:

- It is quite simple to represent a process flow in LEAN.

Model 12

Business and Information Flows - Collaborative

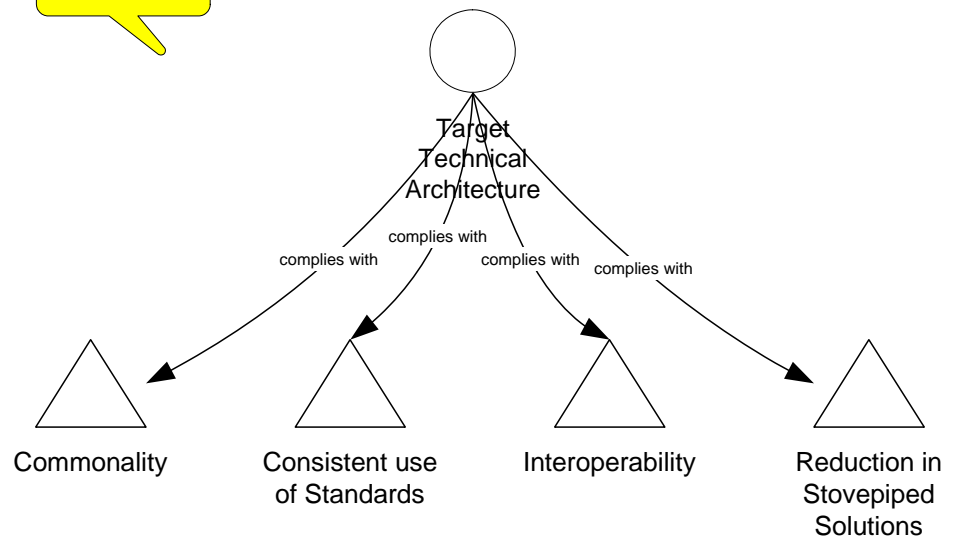


Observations:

- It was a bit more challenging to work out how best to produce this model. As it stands, it conveys the message that the process of collaboration is a part of these other processes. However, Figure 9 also contains the label “Collaborational Groupware”. Is Figure 9 trying to represent the use of Groupware, or the process of collaboration?

Model 13

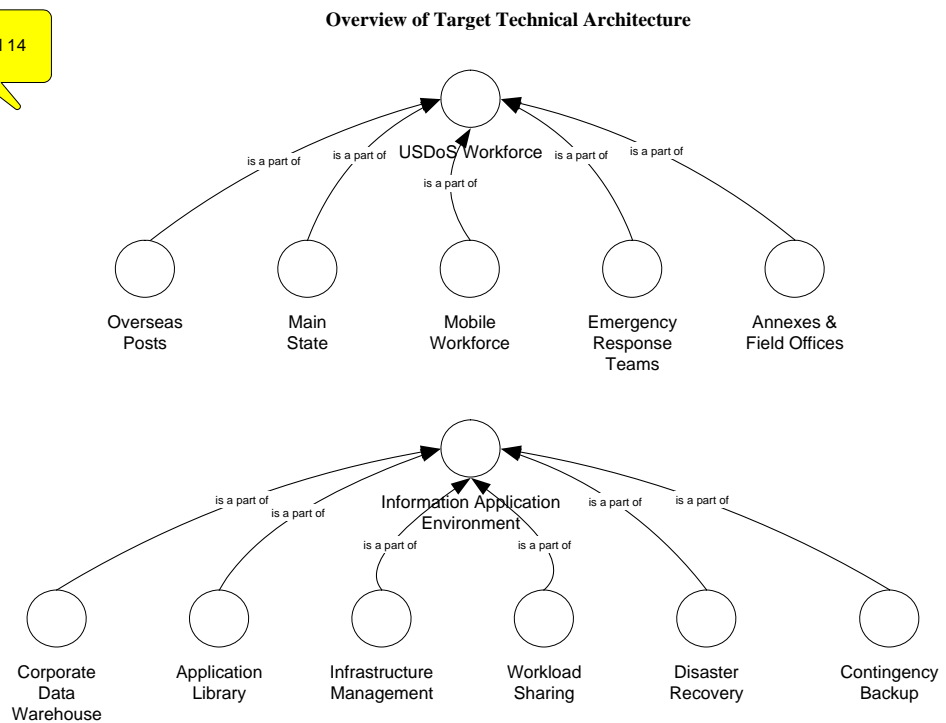
Overview of Target Technical Architecture



Observations:

- Standards, guidelines and principles are important components of EA's. These can be represented as Rules in LEAN making it easy to show how these apply to Resources, Rules or Actions.

Model 14

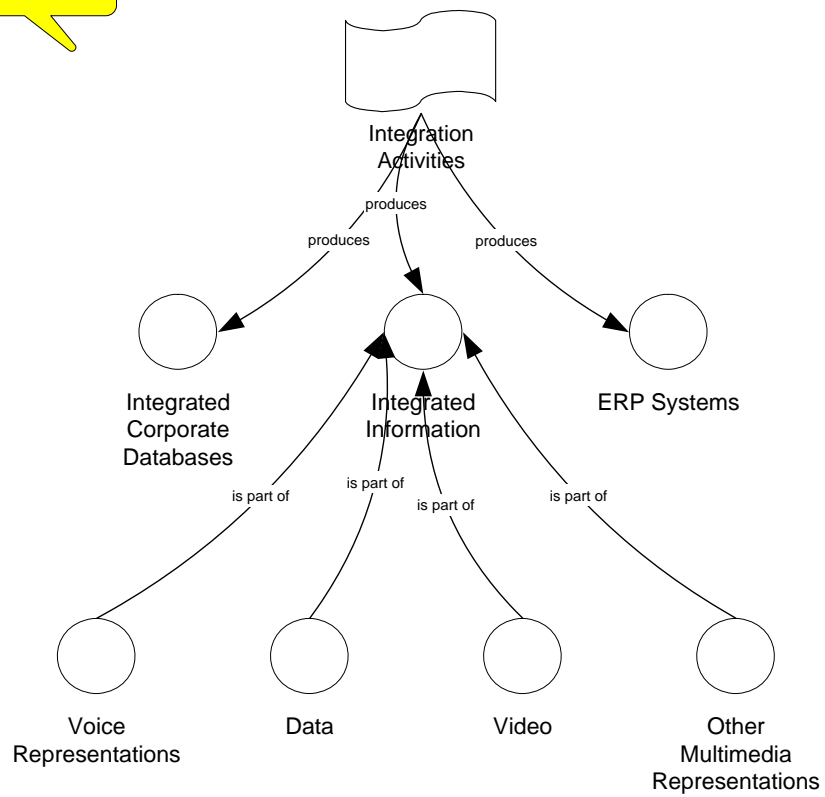


Observations:

- Figure 11 is a relatively complex model. In fact, it combines a picture with a model. The picture is presumably designed to convey the sense that the USDoS operates globally.
- There are a large number of different types of components in Figure 11, including labels, nested graphical structures and isolated graphical components (e.g. “USG Networks”, “Internet”). It is left to the reader to determine how many of these structures relate to each other using prior knowledge. In many cases, the relationships between the various components still remains unclear. In general, Figure 11 represents a set of concepts that are loosely related in some undefined way.
- Since many of the relationships between the components in Figure 11 are undefined, it was very difficult to translate it into a LEAN model. The LEAN model above shows just some components of the original figure. With clarification, a more complete LEAN model could be created.

Model 15

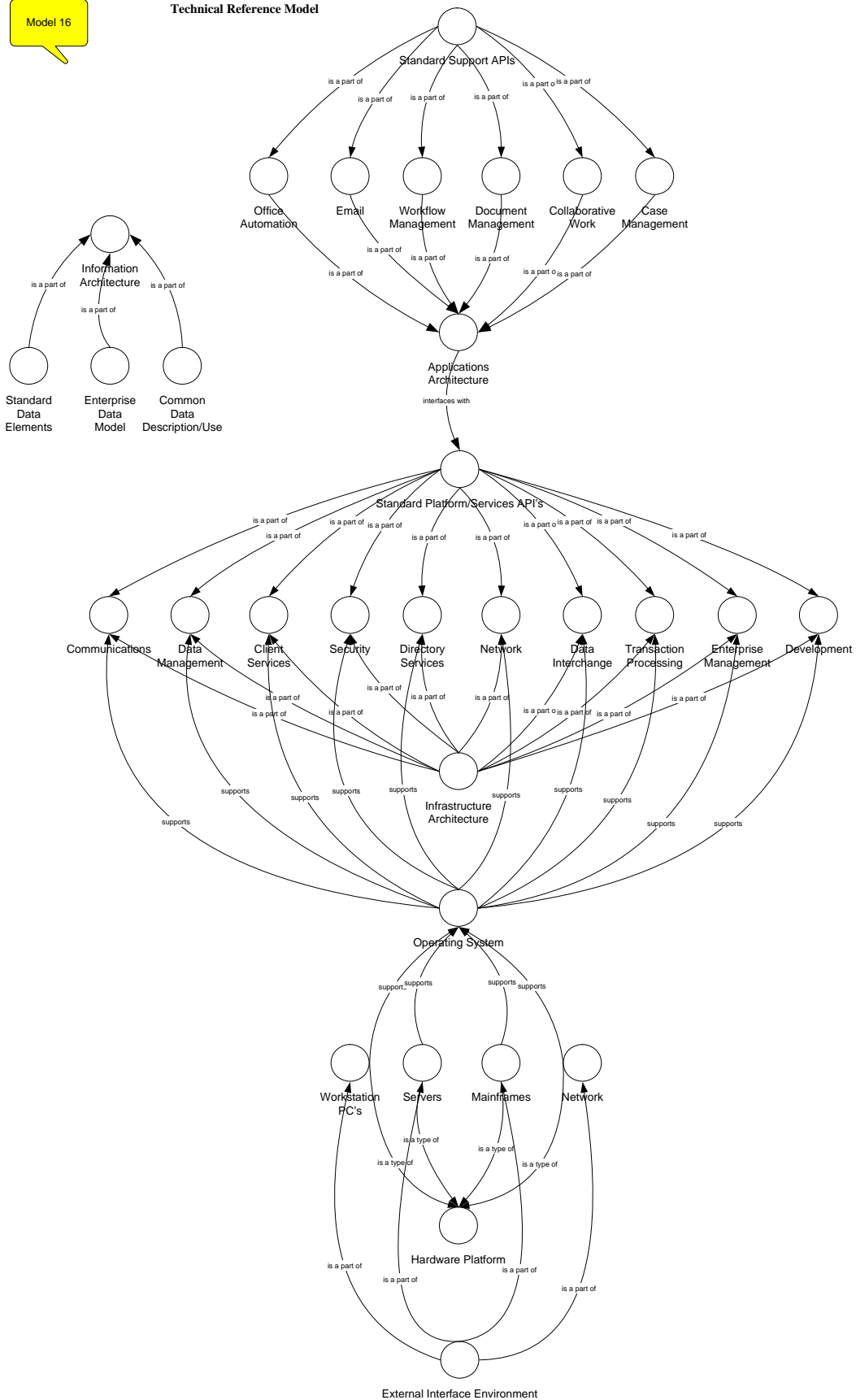
Integration and interoperability



- Observations:
- It is assumed that the future integrated environment will only result from some type of 'integration activity', so this was created and represented as a LEAN Action.
- An information hierarchy was then created to show that the various forms of data will all be integrated.

Model 16

Technical Reference Model



Observations:

- Figure 12 from the USDoS ITA is a typical ‘block model’ as found frequently in EA’s. The block model has some additional labels added in, with their relationships to the blocks undefined. The arrows along the top appear to have no formal meaning.
- As the LEAN translation shows, this block model can be represented largely as a set of Resource hierarchies.
- The concepts of “Support Application Areas” and “Platform Service” were not represented because they would have just cluttered the diagram further. Their representation is a trivial task, resembling the structures already shown.
- Although this block model can be effectively represented in LEAN, the LEAN model appears significantly more complex than the original model. If the semantics governing the block model could be formalised, it may be a preferable way of conveying this type of model.

15 APPENDIX F – DESIGNING AND RE-ENGINEERING SUBSYSTEMS

In this study, we determine whether a highly conceptual metaphor, that is suitable for developing a unified EA modelling language, can also be used to effectively structure an enterprise subsystem. If so, then this adds weight to the assertion that an enterprise metaphor can be used to structure systems, not only at a highly conceptual EA level, but also at lower levels of abstraction. This work was used as the basis for a shorter, published research paper: Elastic Metaphors: Expanding the Philosophy of Interface Design (Khoury and Simoff, 2003).

It will be recalled that one of the challenges of enterprise modelling is that there is often inconsistency between high-level EA models and models at lower levels of abstraction. Part of the reason for this is that these models are usually based on different metaphors. For example, the enterprise model may be based on a ‘city landscape’ metaphor, while the interface to a subsystem within this enterprise is built using a ‘car dashboard’ metaphor.

Subsystems are components of larger systems, which may be part of a *set* of systems that are managed and delivered by an enterprise. In this study, we use a *user interface* as an example of a subsystem. A user interface provides the interface between the user and the ‘backend’ functionality provided by some system. Because user interfaces are graphical subsystems where the attributes of the systems are easily visualised, they serve as an ideal evaluation tool.

Furthermore, the user interfaces selected for this examination are all *email* user interfaces. These were selected for the following reasons:

- A variety of email systems can be publicly accessed.
- The function and operation of email systems is likely to be well understood by the reader.

- All email systems have a set of core functions that are essentially uniform. This is evidenced by the fact that we can use different email systems interchangeably to communicate with one another. This means that we can repeat the test (the redesign of the interface) several times, knowing that in each case, we are dealing with a similar set of functions.
- Most modern email systems provide graphical user interfaces that we can redesign easily and for which the changes in functionality are easily observable.

An email system is defined as a system that allows asynchronous communications between two or more hosts using text and files. It should be noted that the term ‘email’ is a concrete metaphor where the source is a traditional mail system and the target is a system that allows communications over an electronic network.

In this study, one email system was designed ‘ground-up’, and three email systems were re-engineered from existing systems. The new email interfaces were then visually inspected and compared to the original interfaces. This comparison shows whether the application of the theory presented in this thesis has potential efficacy for the structuring of systems through various levels of abstraction.

It should be noted that the design and redesign of these email interfaces were carried out as *logical* exercises. The interfaces were redesigned visually to create prototypes of the new subsystems, but functioning models were not produced. To create four *functioning* email systems would be well beyond the scope of this research, requiring perhaps months, or years, of effort by a team of programmers.

15.1 Developing a Unified Modelling Language

It will be recalled that there are three stages in the methodology presented in this thesis for developing unified modelling languages. These stages are: ‘Identify an enterprise metaphor’, ‘Specify and formalise the language’ and ‘Codify the enterprise metaphor’, as shown previously in Figure 16 - The Applicability of LEAN at Various Levels of Abstraction. We apply this same methodology to the development of a unified modelling language in this section, albeit, less formally since this work falls outside the defined scope of this thesis.

For this exercise, we will use the general concept of a Game as the basis for a unifying metaphor. The metaphor, ‘an enterprise is a game’ fits the criteria we set for unifying EA metaphors described in 4.3 Model Hierarchies. We can assume that most people have a general, intuitive sense of what constitutes a game, even though games can vary enormously in their nature and the potential for new games is infinite. This suggests that the game paradigm is highly conceptual and potentially well suited to our needs. It should also be noted that game metaphors are often used to describe IT systems, and are even been used to describe IT development methodologies (Takeuchi and Nonaka, 1986). Here we use the same concept as the basis for a lower level working metaphor: ‘email is a game’. While this is different in form to the enterprise metaphor ‘an enterprise is a game’, it is essentially the same metaphor: the targets have changed but the metaphor source is the same. Since the same ontology would be used to model at both levels of abstraction, there will be a one-to-one correspondence between these structures that should allow alignment between these different levels of abstraction.

Games are characterised by:

- decision making
- goals
- opposition (or struggle)
- managing resources
- game tokens (through which to effect action)
- information

(Costikyan, 1994)

Users approach an interface with a goal in mind. Goal achievement then involves a struggle to manipulate their tokens appropriately by making the right decisions about how to apply the available resources²⁰. While Struggle and Information are important conceptual aspects of the game metaphor, they are not required as functional components in the following exercise.

²⁰ It is during this struggle that users are likely to become aware of any shortcomings in the interface design.

15.2 Reengineering Existing Interfaces

The Game metaphor is applied in the following stages:

- The existing interface is **decomposed**.
- The required functions are **mapped** onto the Game metaphor.
- The interface is **reconstructed**.

In the following examples, three well-known interfaces are converted to unified metaphor based interfaces. The interfaces used are the Hotmail, Yahoo and Lotus Notes email systems. It should be noted that the choice of these email systems is purely arbitrary and was made based on their availability and popularity. It is expected that the same principles would apply to any email interface.

15.2.1 Decomposition and Mapping

15.2.1.1 Hotmail Interface

Figure 26 shows a partial screen image (some advertisements, browser controls etc have been removed) of the Hotmail interface. This screen is presented once a user selects an item from their “in-box”.

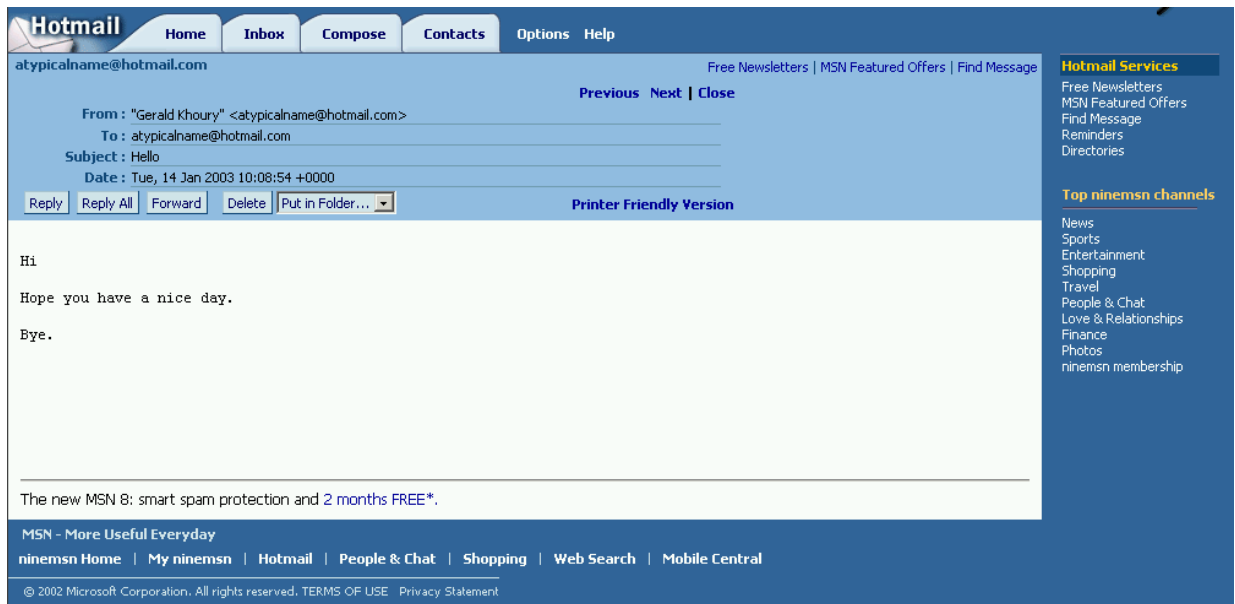


Figure 26 - Hotmail Interface

Figure 27 shows the subset of functions from Figure 26 that will be investigated.

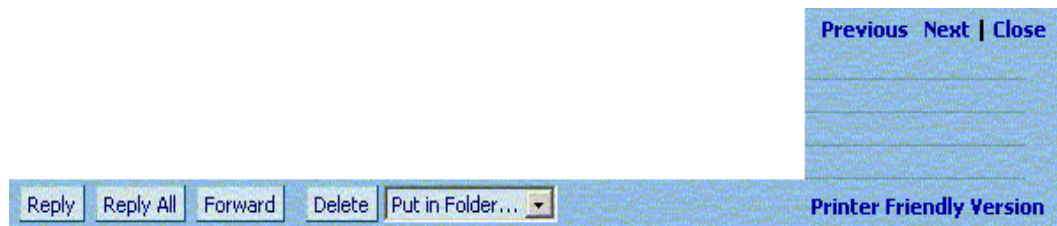


Figure 27 - Hotmail Functions

Many metaphors are in use here: a filing system (folder), container (close), time (previous, next), direction (forward), writing (delete), personal dialogue (reply, version) and human relationships (friendly). Most of these functions are presented using another metaphor: the button metaphor. At a higher level, the entire interface is suggestive of a traditional paper-based mail system (as the name Hotmail implies), and this corresponds to the archetype of the *communicator* (Stefik, 1996). Therefore, not only are many metaphors operating simultaneously; they are also operating at different levels of abstraction. What appear to be simple functions are actually quite complex threads running through a multi-layered web of metaphor (to use another meta-metaphor)!

Unfortunately, the result is an interface that is confusing and inefficient. Yet, this is a simple application with a relatively small set of functions.

Figure 28 summarises the functional decomposition of this interface. The groupings (columns) chosen for this decomposition are based on the game metaphor and thus provide a mapping between the existing interface and the unified metaphor. The decomposition reveals that there are several dimensions to each of the functions under consideration and that each command *implies a set of attributes* (subject, action, variables, values etc). For instance, “Reply” means ‘enable me to *compose* an email to the *person* who sent the email I am *currently* reading’. In terms of metaphor, “Reply” is based on the metaphor of a dialogue: someone has sent me a message and I will now send a message back. The message I send back is *logically* linked to their original message.

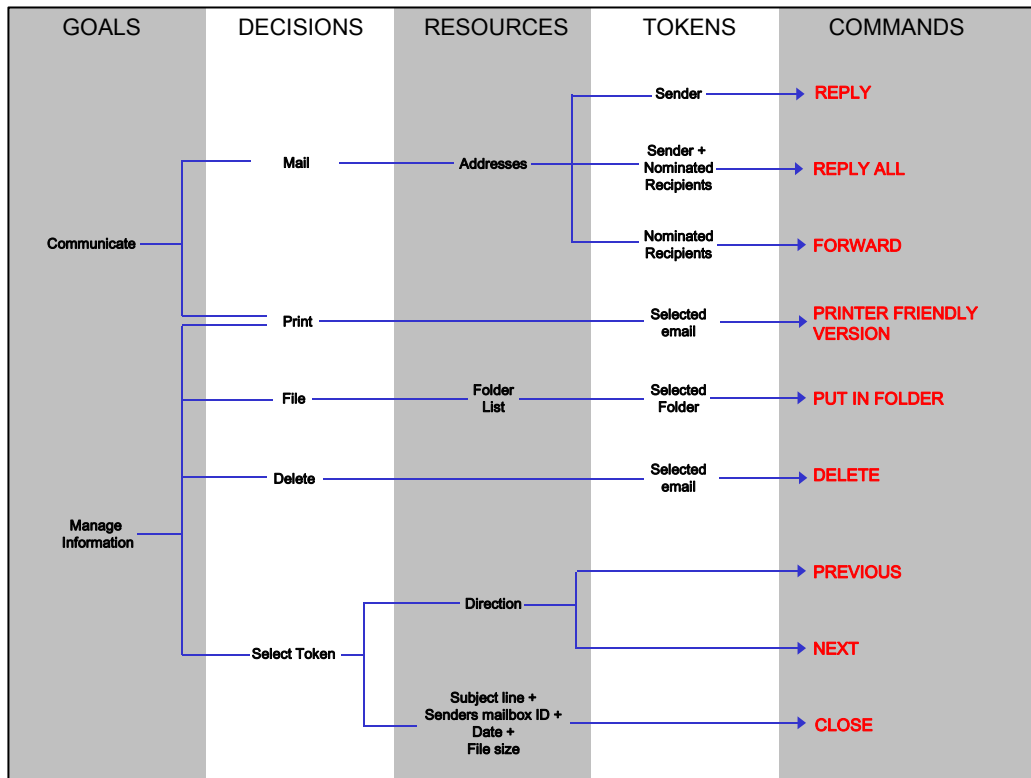


Figure 28 - Decomposition and Mapping of Hotmail Functions

15.2.1.2 Yahoo Interface

Figure 30 shows a partial screen image from the Yahoo mail interface. This screen is

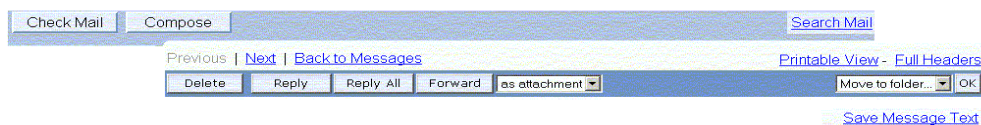


Figure 29 - Yahoo functions

presented once a user selects an item from their “in-box”.

Yahoo takes an interesting approach to usability by duplicating most functions on the screen. In fact, as “Check Mail” and “Back to Messages” perform the same operation, there are four buttons provided to perform this function (actually the “Mail” drop down list provides this function as well).

Figure 29 shows the subset of functions from

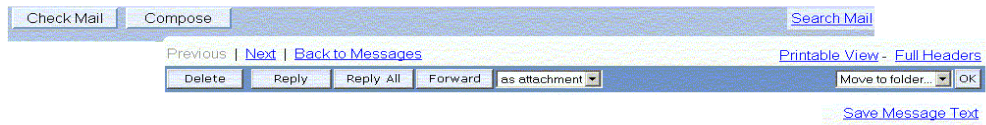


Figure 29 - Yahoo functions

Figure 30 that we will be investigating.

Using the same approach taken for the Hotmail interface, the Yahoo interface is decomposed as shown in

Figure 31.

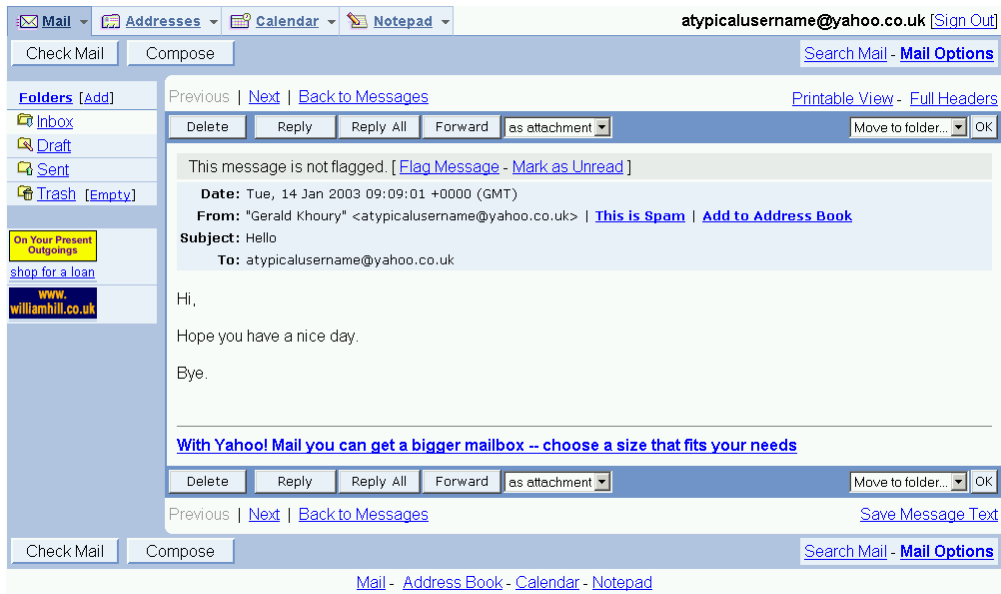


Figure 30 - Yahoo Interface

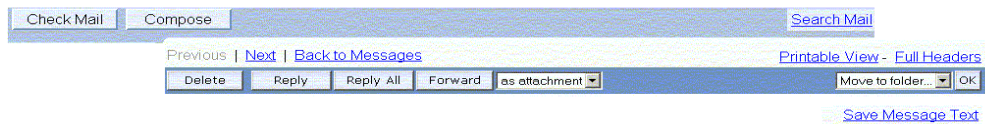


Figure 29 - Yahoo functions

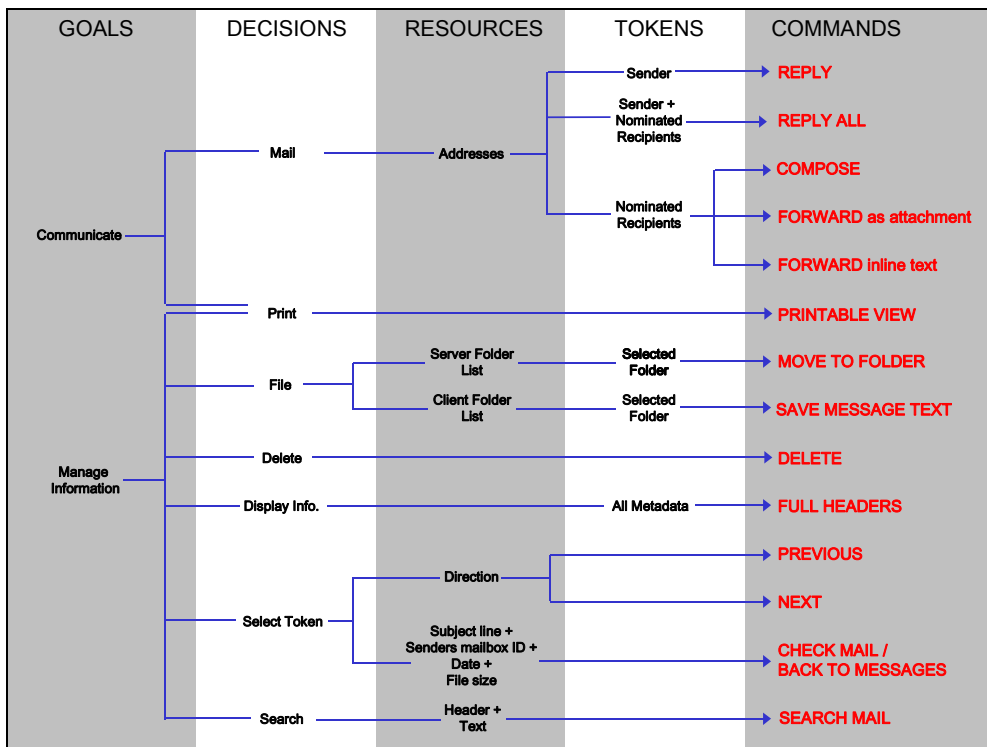


Figure 31 – Decomposition and Mapping of Yahoo Functions

15.2.1.3 Lotus Notes Interface

Figure 32 shows a screen image from the Lotus Notes interface. This screen is presented once a user selects an item from their “in-box”.

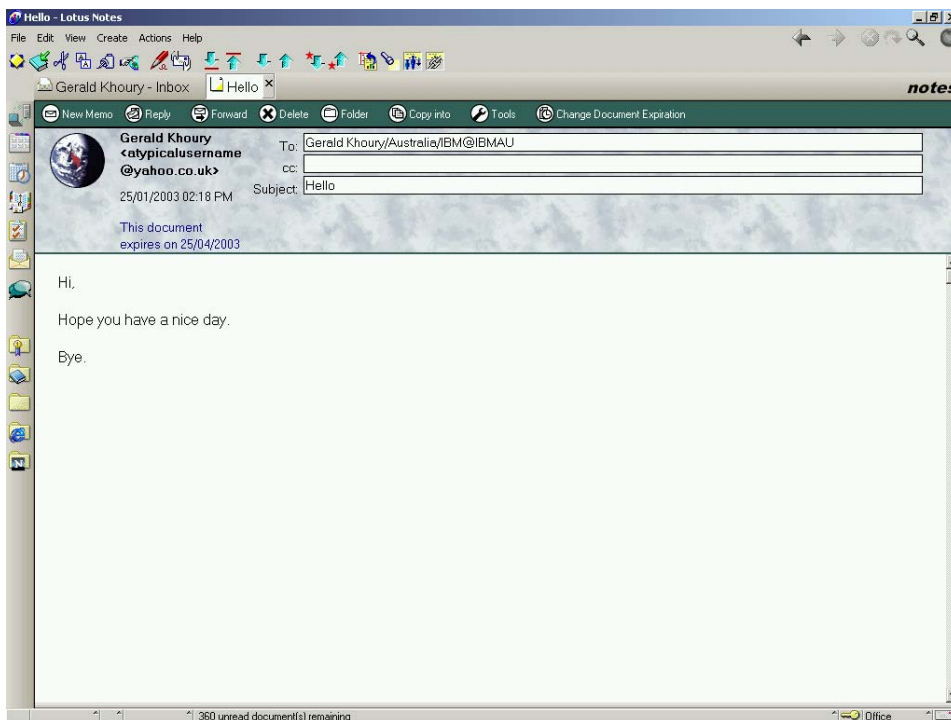


Figure 32 - Lotus Notes Interface

Figure 33 shows the subset of functions that we will be investigating. An interesting aspect of this interface is the use of metaphoric icons to enhance the text buttons.



Figure 33 - Lotus Notes functions

For the sake of brevity, we are only performing a simplified decomposition on a subset of the entire Lotus Notes interface. Many of the functions shown in Figure 33 provide drop down menus with further command options, each of which could be included in this decomposition (and subsequent recomposition using an unified metaphor). Of course, the principles remain the same, but with the analysis extended to the entire interface, the impact of the transformation would be likely to be even more dramatic.

The Tools function seems a particularly anachronistic function. The drop down menu provided by this button is shown in Figure 34.

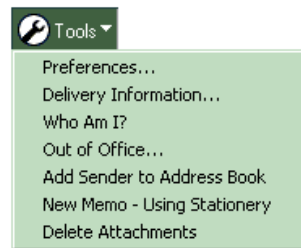


Figure 34 - Lotus Notes "Tools" Drop Down Menu

Two of these sub functions refer to the current email, the rest appear to be an assorted collection of “left over” functions! This set of functions would itself be benefited by an exercise of decomposition and unified metaphor recomposition to determine how they could more logically be organised. We will leave this as a separate exercise for brevity’s sake. The Lotus Notes interface decomposition is shown in Figure 35.

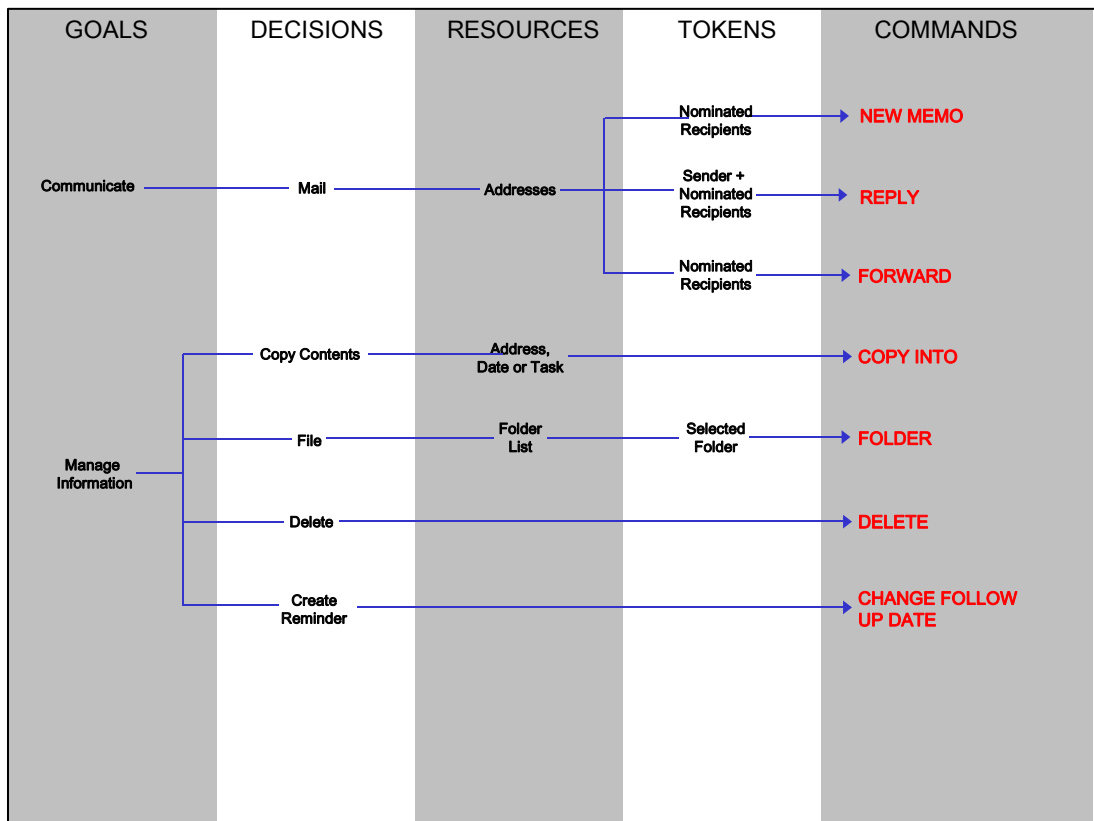


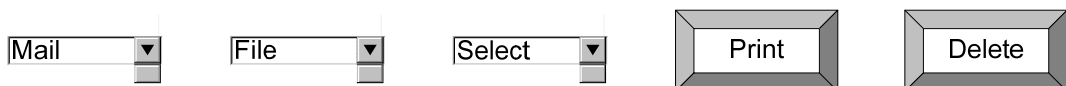
Figure 35 - Decomposition and Mapping of Lotus Notes Functions

15.2.2 Interface Recomposition

In *recomposing* the interface, we need not prohibit ourselves from using concrete metaphors. However, we will impose the restriction that each concrete metaphor will be used to explain a single concept only. This will ensure that there is a clearly defined boundary for every non-unified metaphor.

15.2.2.1 Hotmail Interface

Figure 36 shows the set of functions that is required to represent each of the Decisions shown in Figure 28. These replace the functions used in the original Hotmail interface:



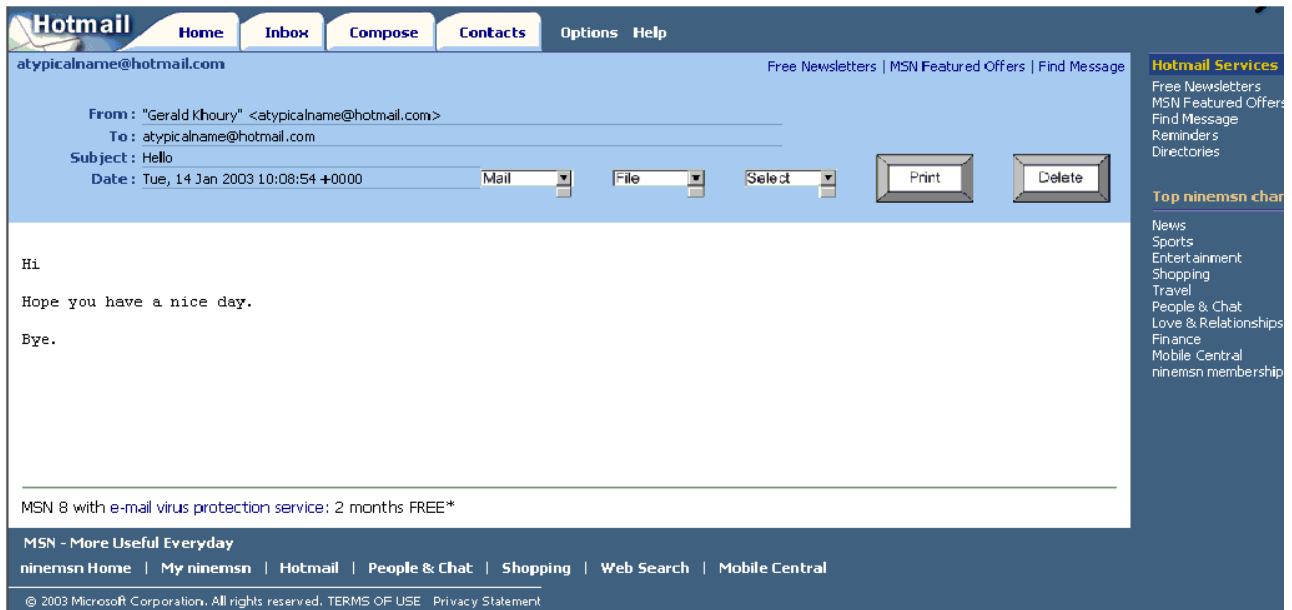


Figure 38 - Recomposition of Hotmail Interface based on Unified Metaphor

Figure 36 - Recomposed Hotmail Functions

Three of the functions shown (Mail, File and Select) are presented as drop down menus, because the decomposition and mapping show that a choice of resources must be made to enact these decisions. Decisions to Print or Delete have no associated resources. If decisions are made to Mail, File, or Select a new token, then the resources shown in Figure 37 are offered:



Figure 37 - Recomposed Hotmail Functions - Expanded

Thus, the reconstructed interface would look similar to Figure 38:

Comparing this to the original interface shown in Figure 26, we see that nine initial interface functions have been replaced with five new functions. The mapping is as follows:

- Mail = Reply, Reply All, Forward
- Delete = Delete

- File = Put in Folder
- Print = Format for Printer
- Change = Previous, Next and Close

No functionality has been lost.

15.2.2.2 Yahoo Interface

Figure 39 shows the set of functions that is required to represent each of the Decisions shown in Figure 31. These replace the functions used in the original Yahoo interface:

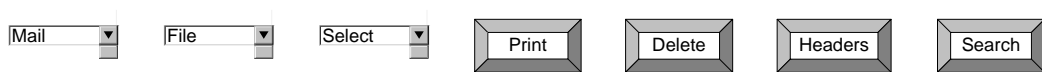


Figure 39 - Recomposed Yahoo Functions

Although Search does have associated Resources to choose from, it is not presented as a drop down list because Yahoo provides a separate screen where the search terms can be entered. If decisions are made to Mail, File, or Select a new token, then the resources shown in Figure 40 are offered:



Figure 40 - Recomposed Yahoo Functions - Expanded

If Mail – New is selected, another set of choices would be provided to allow the user to choose to Compose a new message, forward the current message as an attachment, or forward the current message as inline text. Thus, the reconstructed interface would look similar to Figure 41:

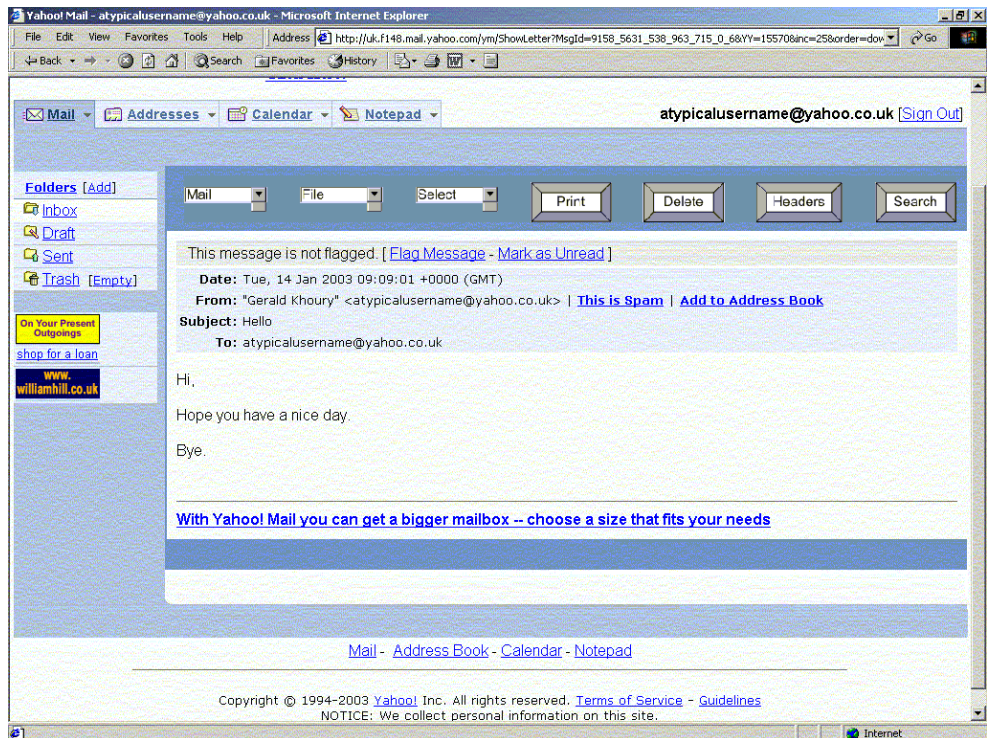


Figure 41 - Recomposition of Yahoo Interface based on Unified Metaphor

In this case, thirty-one screen elements have been replaced with seven! Despite the radical simplification of the interface, no functionality has been lost.

15.2.2.3 Lotus Notes Interface

Figure 42 shows the set of functions that is required to represent each of the Decisions shown in Figure 35. These replace the functions used in the original Lotus Notes interface:



Figure 42 - Recomposed Lotus Notes Functions

If decisions are made to Mail, File, or Copy a token, then the resources shown in are offered Figure 43:

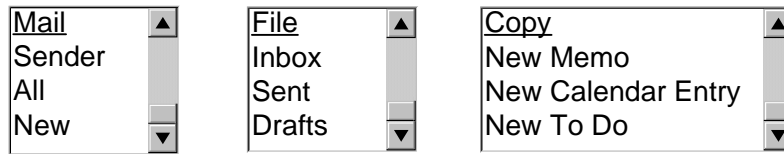


Figure 43 - Recomposed Lotus Notes Functions - Expanded

Thus, the reconstructed interface would look similar to Figure 44.

15.2.3 The Result

The methodology applied here has allowed us to replace a haphazard collection of functions with a concise set of functions that is aligned with the way users use interfaces. That is, users make decisions on how best to use the available resources in order to achieve goals.

This has been achieved by decomposing the original interface, a translation to the chosen unified metaphor (in this case, a Game metaphor) and then recomposing the interface.

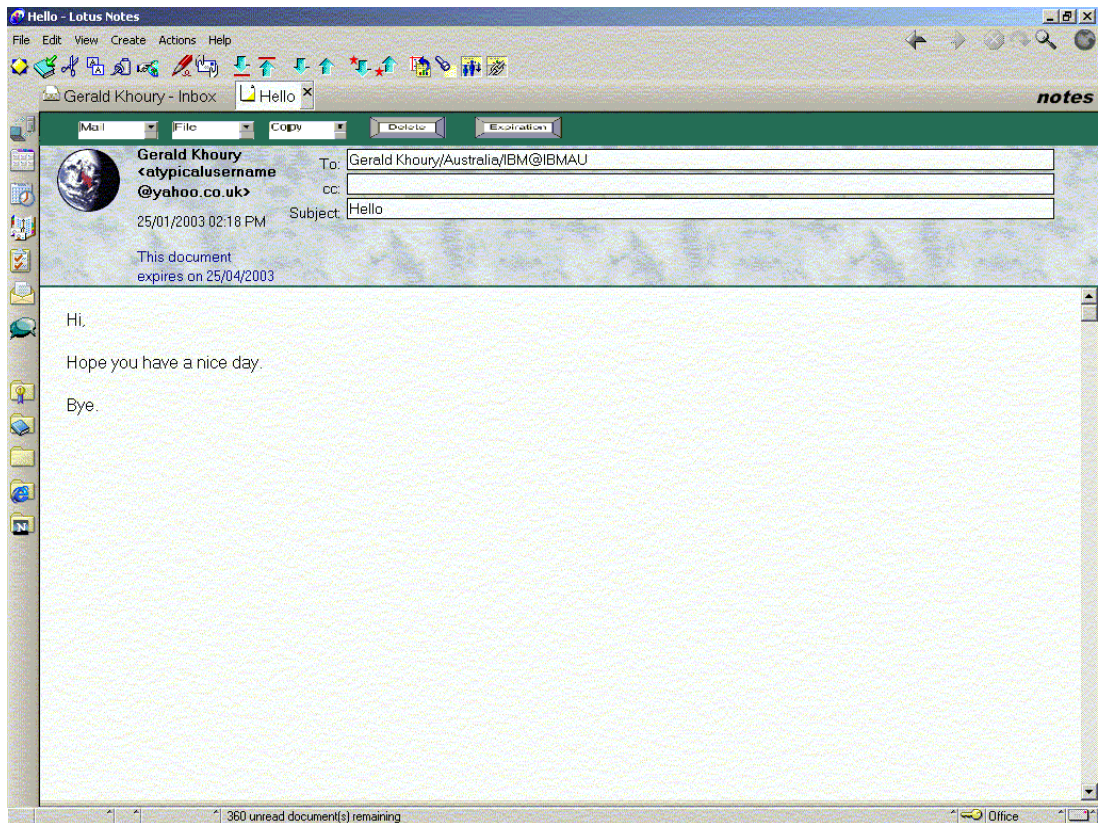


Figure 44 - Recomposition of Lotus Notes Interface based on Unified Metaphor

The result is an interface that is much more attuned to the way humans experience the world, based upon a consistent and coherent metaphor that can be applied to any interface.

15.3 Developing New Interfaces

This approach can also be used to develop *new* interfaces. In this situation, the unified metaphor is applied in the following stages:

- **Identify** the required functions.
- **Map** the required functions onto the unified metaphor.
- **Construct** the interface.

In the following example, a new, email system is constructed.

15.3.1 Identifying the Required Functions

As our previous analysis of email systems shows, the commonly provided functions of an email system include:

- Mailing messages
- Filing messages
- Deleting messages
- Printing messages
- Selecting a message from a list

15.3.2 Mapping

The required functions can be mapped onto the game metaphor as shown in Figure 45.

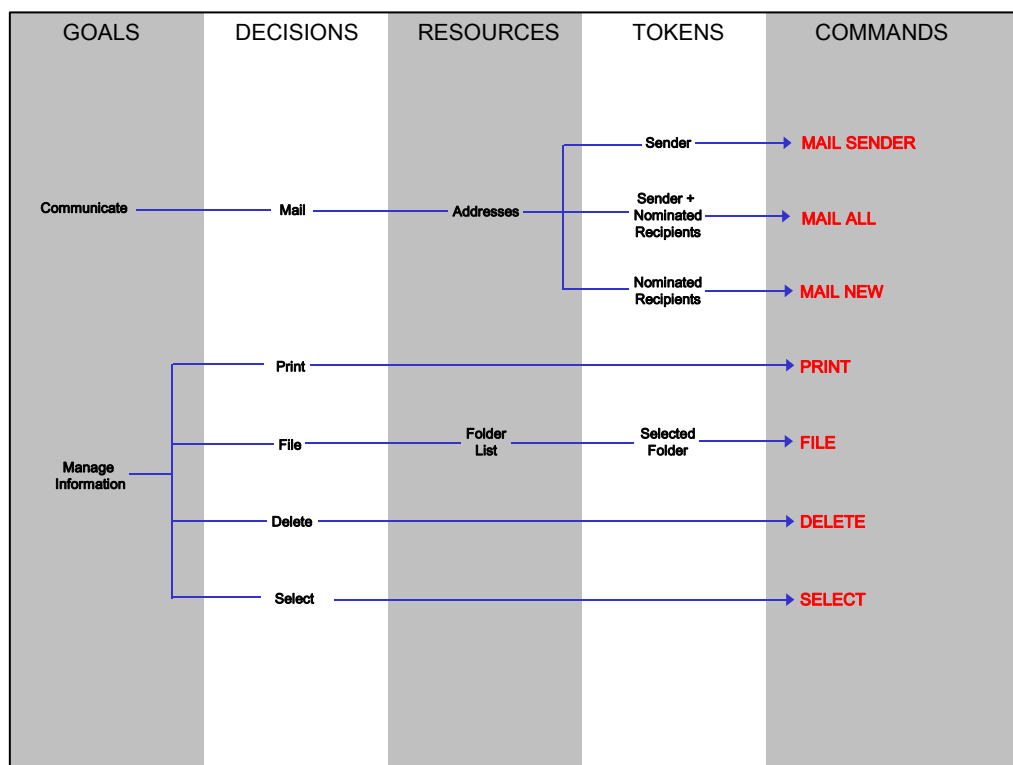


Figure 45 - Mapping of New Email Interface Functions

15.3.3 Construction

Figure 46 shows the set of functions required to represent each of the Decisions shown in Figure 45. These functions would be provided as part of a new email interface.

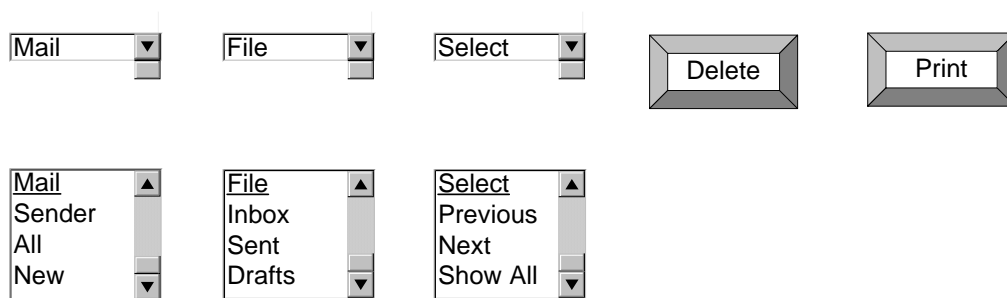


Figure 46 - New Email Interface Functions

15.4 Conclusion

It has been shown that a metaphor that is unified at an enterprise level can also be used to structure systems at a much lower level of abstraction. If it can be demonstrated that such a metaphor can be applied at *all* levels of abstraction, then we will have shown that a unified metaphor can be used to structurally align the implementation of systems with the high-level strategic design of an enterprise. This provides a foundation for future research.

It has also been shown that the use of a unified metaphor may offer advantages over contemporary metaphorical approaches to systems development. A significant advantage purported through the use of a unified metaphor is the flexibility and expandability offered. When the first email systems were developed, synchronous communications (e.g. instant messaging) were not part of the solution being offered. Thus, the term “email” appeared rather appropriate, as there is a strong analogy between asynchronous messaging and traditional mail.

Instant messaging has now become popular, but the email metaphor does not readily accommodate it. This is perhaps the reason why suppliers offering both email and instant messaging services usually provide them as separate products. By building the email interface using a unified metaphor, we have developed a system that can easily be

expanded to incorporate other forms of human communications such as instant messaging, groupware, game playing etc.