ONTOLOGY-ORIENTED E-GOVERNMENT SERVICE INTEGRATION UTILISING THE SEMANTIC WEB

Farzad Sanati

A thesis submitted for the degree of
Doctor of Philosophy

University of Technology, Sydney

July 2011

Copyright © Farzad Sanati, 2011 all rights reserved

Certificate of Authorship/Originality

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text. I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Student

Production Note:
Signature removed prior to publication.

____

Acknowledgements

This thesis is the result of a four years effort to climb the mount improbable of my life. It has given me the possibility to engage in even higher challenges, which there are many ahead. Looking back to my physical journey from my birth land Kurdistan to Australia and from who I was to whom I have become, I realise that I could have never completed it without the help of so many people. This is the time to thank them all!

I would like to express my sincere gratitude to my principal supervisor, Professor Jie Lu, for her continuous encouragement, advice, help and invaluable suggestions. She has been a generous, helpful and kind hearted person. Many thanks also to my co-supervisor, Professor Guangquan Zhang, for his valued suggestions to this study.

I wish to thank my fellow research students in our Decision Systems and e-Service Intelligence (DeSI) and the staff of the Faculty of Engineering and Information Technology, University of Technology, Sydney (UTS) for their various assistance and advice are of great benefit to this study. I appreciate the financial support from both the Faculty of Engineering and Information Technology and the Centre for Quantum Computation and Intelligent Systems (QCIS).

I appreciate the travel support for attending the international conferences which I received from the Faculty of Engineering and Information Technology, the UTS Vice- Chancellor’s Conference Fund and QCIS. This thesis received editorial advice from Ms Sue Felix, helping to identify the correct grammar, syntax and presentation problems.

I would like to express my heartfelt appreciation and gratitude to my parents for all their love and support, also my brothers and sisters for their constant encouragement over all these years.

The Last but certainly not the least, I am extremely fortunate that I could share the joy and the pain of the last four years with my dear wife, Baharak. She has been by my side during hard times and comforted me and encouraged me to continue. I am grateful for her love and endless optimism in times when barriers seemed impossible to pass. Shano and Kani this is also for you to remember that no matter how improbable things may look, nothing is impossible.

Table of Content

Table of Figures

List of Tables

Table of Listings

Abstract

E-government service integration process has recently become an important research topic in e-government domain since many countries have developed various levels of e-government services. Non-interoperability between government agencies in service delivery implementation and platform posing the technical challenge, and the lack of the formulated modelling framework is the main methodological obstacle on the way of achieving dynamic delivery of integrated e-government services.

This research is a study of the problems associated with the integration and delivery of integrated e-government services, and proposes a novel solution to tackle them. We start with investigating the fundamentals of e-government as a field of research to build a sensible argument for the questions investigated by this research, which lead to the exposure of the methodological as well as technological problems with the mechanics of e-government in the areas of service integration and delivery.

The outcomes of this study in Chapters 3, 4, 5, 6, and 7 respectively 1) suggests the most practically relevant and technically possible evolutionary pathway to e-government transformation, 2) proposes a modified software engineering process to achieve such transformation, 3) develops an innovative framework for modelling the service integration, 4) proposes an ontology as its knowledgebase, and 5) develops an innovative and intelligent software to support the practice of service integration and delivery. These outcomes collectively result in the introduction of a novel, complete and coherent solution for the abovementioned problems.

This research is a cross disciplinary study of software integration engineering frameworks, e-government service delivery platform and semantic web technology, all working to devise the most efficient and robust framework of using semantic web capabilities to enable the delivery of integrated e-government services in an intelligent platform.

Chapter 1: Introduction

1.1 Background

E-government is defined as:

The use by government agencies of information technologies (such as Wide Area Networks, the Internet, and mobile computing) that have the ability to transform relations with citizens, businesses, and other arms of government. (The-World-Bank, 2007)

Rapid growth in Information and Communication Technology (ICT) has substantially dominated the ordinary citizen's everyday life. One visible extension of this change is measured by the recent surge in public administration service delivery in the Internet domain. In recent years, countries of European Union, North America and many Australian government agencies have developed extensive e-government services for individual citizens as well as for private businesses.

This rapid growth in e-government services has resulted in duplicate and redundant services that could lead to confusion for customers and add complexity to conduct business with government when more than one government agency is involved. Customers who want to deal with government must first discover which part of government provides their desired services, and this problem can become even more complicated if there are multiple agencies involved, in which case the problem of interoperability and incompatibility of these services would add to the already complicated equation.

Early steps towards ICT interoperability were taken in the early 1980s when standardisation in ICT started as a typical response to concerns related to proprietary systems (Guijarro, 2007). Since then, standardisation has come a long way in easing the burden of connecting disparate and incompatible systems by introducing many engineering protocols.

The Australian Government has made fundamental changes to the way it works since the late 1990s under the influence of the rapidly growing capabilities of the Internet. This Government has started the implementation of a long-term plan to achieve a unified and integrated web presence for all its agencies across all levels of government (IMSC, 2007). The integrated government e-services initiative was intended to eliminate redundant systems and significantly improve the Government’s quality of customer service, as well as to generate a dramatic increase in the number of services provided to citizens over the Internet. Integration of e-government services would result in the creation of composite services, increasing greatly the number of information resources and services available to the public through individual agencies. Considering such an ambitious target, the semantic and technical interoperability of integration solutions remains the main problem. This research is committed to proposing a solution to interoperability problems for e-government service integration and delivery.

This chapter is an overview of the research conducted in this area. It helps readers to obtain a cohesive picture of the research conducted in this study and consists of four main sections. Section 1.1 is the problem analysis of this study and explains the research issues that this study attempts to tackle. Section 1.2 is a brief description of the contribution of this study to the body of knowledge in the area of delivering integrated e-government services to services users. Section 1.3 illustrates the overall structure of this thesis. Finally, Section 1.4 offers a list of the author's publications related to this study.

1.2 Research Questions

Government policies lean towards encouraging the use of the Internet by agencies to promote the efficient delivery of all appropriate services and programs to citizens and other services users. These policies have led to considerable investment in Internet-based service delivery by agencies. Based on a preliminary investigation and the literature review, the following problems have been recognised from the study of existing Government to Citizen (G2C) service delivery systems, and research recommendation is extracted for this study.

In order for e-government service integration to become a success it has to handle the following issues:

1) Incompatible interagency process and workflows,
2) Unfamiliar semantics,
3) Complex inter-agency regulatory rules,
4) Development of duplicated and redundant services, and
5) Automatic composition and delivery of services.

A higher level of intelligence in service delivery is a highly desired attribute of e-government systems that could add great value to such systems. Under the guidance of the Australian Federal Government (IMSC, 2007), agencies have launched personalised information delivery services;

however, these efforts are not in line with a unified structure that is planned for the maximum efficiency of government future integration projects. Therefore, this study recognises that to resolve the five aforementioned issues, it is important to focus on, and find answers to, the following research questions:

Question one: What is the most intelligent framework for e-government service integration and delivery?

Recent efforts by public administration to transform itself have resulted in drastic changes in public service delivery mechanisms. Moving towards an electronic means of delivering services was the most important step in the creation of a new paradigm in public service delivery called 'e-government'. Integrated e-government is expected to bring increased efficiency, better and more available services and increased participation by both government agencies and citizens. Citizens in particular must be able to access e-government integrated services effortlessly with minimum working knowledge of their operation.

Integrated delivery of e-government services is one of the most important steps towards ubiquitous government; however, there is no proven framework to develop and deliver e-government integrated services in a coherent, repeatable way. All e-government service integration efforts investigated by this research seem to have been done on a case-by-case basis, without a unified methodological framework (Mugellini, 2005, Guidi et al., 2007) that could be applied to different government systems.

In this research, we pay special attention to the mechanics of e-government service delivery, and within this topic we highlight the lack of a formulated modelling and implementation framework as the main methodological problem in achieving seamless delivery and integration of e-government services. The main focus of this study, then, is to investigate the integration of e-government at Service Delivery Level. This problem is closely related to issues 1, 2 and 4 mentioned above.

Question two: What and how can semantic web technology be used to overcome the technical difficulty of automatic integration and delivery of e-government services?

Service Oriented Computing (SOC) (Munindar et al., 2006) is a software architecture domain that defines a framework to enable building new software applications from existing building blocks (web services). Achieving the best outcome in the integration and delivery of e-government services would require a facility capable of composing web services, not only from one service provider but from different locations and providers. This is a highly complex task and it is already beyond human capability to deal with the whole process manually.

The traditional workflow design paradigm relies heavily on humans who specify the business processes in design time. However, such a manual design approach is not suitable for many cases such as:

1) Inter-agency workflow that crosses autonomous organisational boundaries and requires experts who possess the knowledge required for defining workflows from the constituent organisations, and
2) Customised workflows that might suit individual service users, which would require many variations and make it infeasible and error-prone to predefine the complex workflow in advance (design-time).

Researchers have often suggested the use of semantic web technologies to manage and dynamically configure the composition of web services. Dynamic configuration of integrated web services will enable integration applications to use ontology as an information source to create personalised alternative workflows to meet the requirements of specific service users (citizens) or in response to changing customer preferences in run-time. It also can be a fatal problem for composite service workflows if one or more services fail at run-time. This problem is closely related to issues 1, 2, 3 and 5.

1.3 Research Objectives

Achieving repeatability is perceived to be one of the most fundamental attributes of any framework in any engineering discipline. Many research projects are currently attempting to formulate modelling strategies utilising new technologies and development techniques proposed in the Service Oriented Architecture (SOA) (Erl, 2005) domain. In the absence of a documented best common practice of design and modelling techniques for e-government service integration, it would be greatly beneficial to discover such common practice and describe the detailed specifications of a repeatable modelling framework for such projects. At the same time, the absence of an intelligent delivery platform for integrated e-government services is the other significant problem concerning the concept of e-government service integration and delivery. Hence, the main objectives of this study are recognised as follows:

1) To investigate and recognise the actual e-government evolutionary trend and propose the most practical and realistic transformation strategy for e-government.
2) To investigate currently proposed e-government service integration models in order to scope the problem domain, including service integration practices and the classification of e-government service integration strategies. This is essential to discover the implementation and technical strategies of current solutions,
3) To propose detailed specifications of a repeatable modelling framework for the delivery of e-government integrated services. This needs to be a repeatable modelling process, clearly defining all the tasks and activities needed to effectively model integrated service workflows and having more effective management control over all development activities, and
4) To select the technology and develop the most efficient knowledgebase component for the intelligent automation of the e-government service integration and delivery system.
5) To propose a delivery platform for integrated e-government services. This prototype application needs to make use of semantic web technologies to create and manage personalised configuration for composite e-service workflows. The platform will be used to deploy and deliver integrated e-government services in general.

1.4 Significance

Government institutions across the world, at national, regional, and local levels, are significant consumers of technology. Governmental services affect us all. They can range from areas of defence and national security to health, taxation, law enforcement, judiciary, environment, energy, social services, disaster management, and land use management.

Web services technology has been embraced by industry and governments as the new standard for enabling an organisation to transact with its partners over the Internet in support of its overall business strategy. IT culture within the public sector has long been known to be unique. The responsibilities of managing a wide range of often critical public services establish a distinct set of priorities that cannot be compromised, especially when it comes to reliance on technology.

Web services technology adoption has grown substantially in government agencies in federal, state, and local sectors. There is an increasing realisation that the strategic benefits of service-orientation can help overcome many of the traditional cost and efficiency-related IT problems (Juneja et al., 2008).

The outcome of this study will significantly improve the current position of e-government service delivery practices by introducing an effective and relatively easy-to-follow framework for the integration and delivery of e-government services. This study will provide an innovative design for e-government service integration and delivery workflow management, using intelligent design to handle dynamic workflow composition and failure recovery at run-time, such that it would enable ordinary citizens to create personalised composite service and execute them to achieve desired legal outcomes. It makes use of the semantic capability of ontology to achieve this objective.

Semantically transparent e-services make it possible for clients to successfully use services that are dynamically discovered without prior negotiations between the workflow composer (client) and service providers (government agencies). Such goals are important for web service environments, including G2C as well as Government to Business (G2B) applications (Burstein et al., 2005).

1.5 Contributions

This research makes the following contributions to the theory and practices of e-government service integration and delivery:

1) Present arguments to establish a theory for the proposal of the most practically relevant and technically possible evolutionary pathway to e-government transformation, based on the body of evidence presented in Chapter 2
2) Proposal of a modified software engineering process to deal with the specific tasks and activities of service integration projects. This new engineering process is called Service Integration Engineering (SIE) and is designed to handle the complexity of developing integrated e-government services.
3) Development of an innovative E-government Service Integration Modelling (ESIM) framework, recognising that the ability to offer a citizen-centric view of government model is the key to a successful e-government service delivery. We adopt the LifeEvent model which is the most widely-adopted paradigm supporting the idea of composing a single complex e-government service that corresponds to an event in a citizen's life. In our interpretation of LifeEvent, elementary building blocks of LifeEvent are web services offered by multiple government agencies or other business partners. This

research defines ESIM as a framework based on ontological analysis and modelling that makes extensive use of the LifeEvent concept. ESIM is a top-down abstraction approach in requirements elicitation and modelling to define and implement the design of LifeEvent in the context of e-government service integration and delivery.

4) The conceptual design of LifeEvent upper ontology. This ontology will provide knowledgebase support for the implementation of the LifeEvent Ontology Oriented Integrated Services platform within the e-government domain. LifeEvent ontology is a logical extension of Ontology Web Language for Services (OWL-S); we are interested in extending this existing approach to enable the design of a LifeEvent Metamodel. This Metamodel is to be used as an alterable workflow for a composite service. The resulting ontology covers specific web services semantic concepts to implement the phenomenon of LifeEvent in the context of e-government by:
   a) facilitating the construction of alternative integrated service workflows from entirely different web service vendors;
   b) enabling the repair or re-configuration of LifeEvent workflows in runtime;
   c) the invocation of web services according to the workflow sequence and user preferences at run-time.

5) LifeEvent Ontology Oriented Integrated Services (LOOIS) platform is the fifth contribution of this study. It is a prototype software system for delivery of personalised LifeEvent to customers. This platform will make use of semantic web technology as the

knowledgebase of its core functionality and will validate the modelling assumptions of the ESIM framework. LOOIS will make use of enhanced semantic web technology design techniques specialised to achieve more intelligently manageable dynamic integration of elementary e-government services from multiple sources.

1.6 Research Methodology

The research methodology applied in this study is a set of methods, procedures and tools to conduct research in a certain domain (Nunamaker and Chen, 1990). A number of research methodologies are proposed, among them case study, design research, field study, experimental, and action research. This research will apply design research methodology (AIS, 2007) to achieve the desired objectives of this research.

1.6.1 Design Research

The focus of design research is on preparing and analysing artefacts in order to gain insight into research problems. Based on the principles of this model, this study plans to achieve its objectives in the five steps illustrated in Figure 1.1.

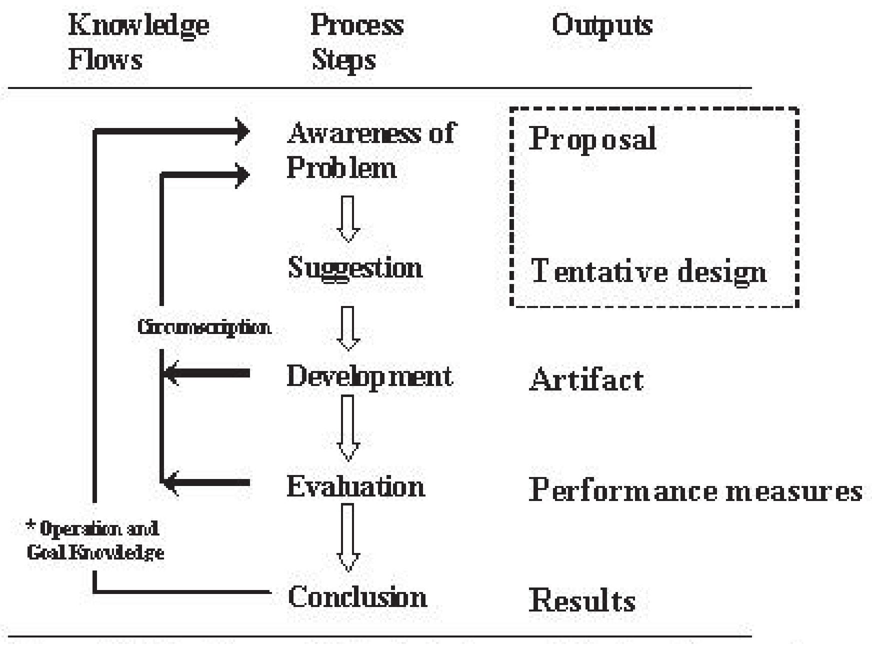


Figure 1.1: General methodology of design research (AIS, 2007)

Awareness of Problem: This is the starting point of the design research, in which we focus on the investigation of current general e-government service applications and development, and in this investigation, meaningful research problems are identified. A survey-based approach has been used to investigate the state-of-the-art and characteristics of the current software development industry. The survey is designed to evaluate and compare the trends in academic literature with current industry practices. We were particularly interested in discovering the tools and methodologies used in web services delivery. In this step, a sufficient quantity of related literature is reviewed to clarify the research questions. The areas of review address e-government in general, software engineering process, e-service integration, semantic web, ontology, and knowledgebase systems. In this step, research problems are clarified and defined using broad literature review and also a survey of the industry practice. The research problems must clearly reflect a gap between existing applications and expected status. The corresponding output of this step is a research proposal.

Suggestion: Following the identification of research problems, a research method process is suggested. The research process describes what the intended artefacts will be and how they can be developed. Suggestion is a creative step towards new functionality foreseen either as a novel configuration of an existing element in a system or adding elements. This research has applied a method of system prototype development in order to validate the theoretical assumptions and concepts.

Development: Artefacts developed in this step testify to the feasibility and reasonability of the original design. The artefacts are built in an iterative process wherein an initial prototype evolves in an incremental manner as the researcher gains a deeper understanding of the research problem. The techniques for implementation will vary according to the artefact to be constructed. The knowledge obtained in this step is fed back to the previous step in order to revise the design and the proposal.

Evaluation: Once constructed, the artefact is evaluated according to criteria that are always implicit and frequently made explicit in the Proposal. Performance of the developed artefacts is evaluated according to the criteria defined in the research proposal and must confirm to the suggested design. These results are then fed back to the previous two steps to improve the design and consequently the artefacts.

Conclusion: The final task of this research is to reach a conclusion as to whether the results are satisfactory. There might still be deviations between the proposal and the developed artefacts; however the conclusions are still valid if the artefacts are considered ‘good enough’. This phase is the final stage of a specific research effort. Typically, it is the result of satisfying, that is, although there are still deviations in the behaviour of the artefact from the multiply-revised hypothetical predictions, the results are judged to be ‘good enough’. Not only are the results of the effort consolidated and written up in this phase, but the knowledge gained in the effort is frequently categorised as either ‘firm’ - facts that have been learned and can be repeatably applied or behaviour that can be repeatably invoked - or as ‘loose ends’ – irregular behaviour that defies explanation and may well serve as the subject of further research.

1.6.2 Research Process

We used design research methodology to prepare the research process. The complete process of this research consists of nine steps divided in to three phases (see Figure 1.2).

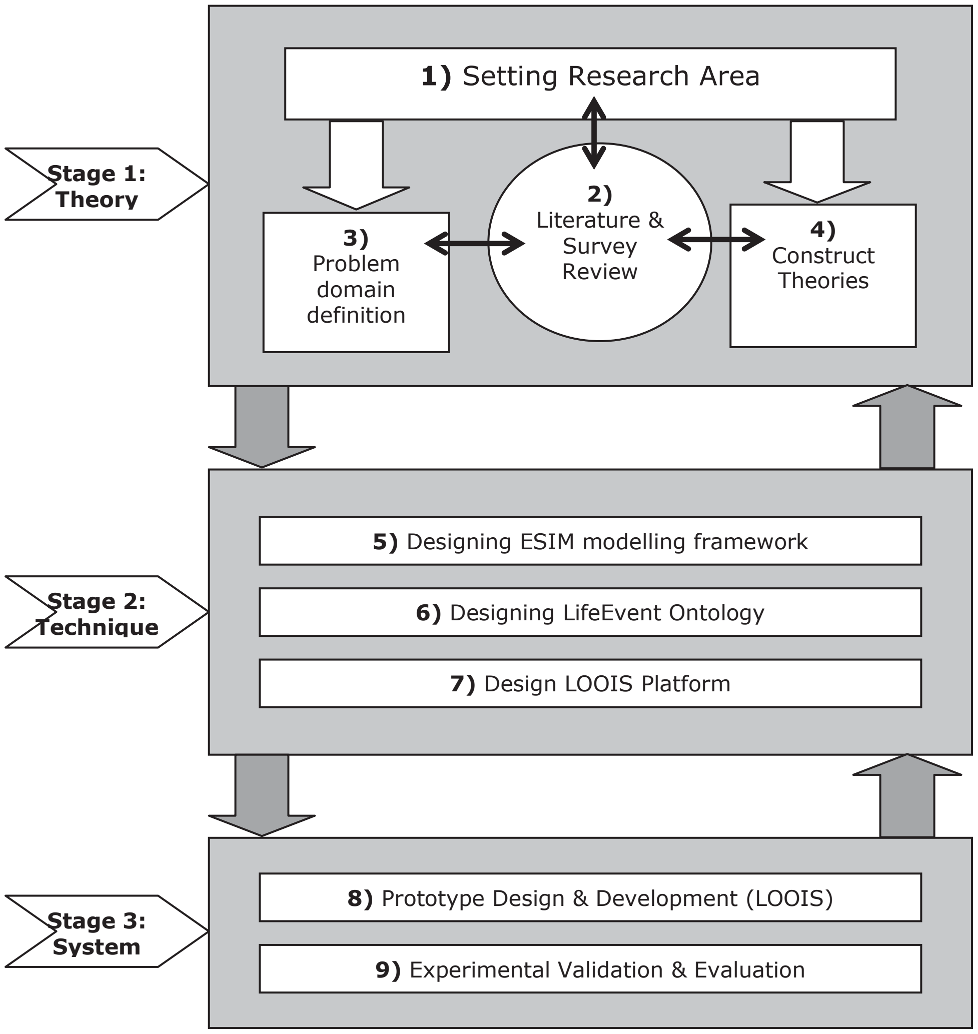

1) Setting Research Area
2) Literature & Survey Review
3) Problem domain definition
4) Construct Theories
Stage 1: Theory
5) Designing ESIM modelling framework
6) Designing LifeEvent Ontology
7) Design LOOIS Platform
Stage 2: Technique
8) Prototype Design & Development (LOOIS)
9) Experimental Validation & Evaluation
Stage 3: System

Figure 1.2: The research process

(Phase 1 – Theory)

There are four steps in this Phase, which allows for carrying out major theoretical work in relation to this research.

- **Step 1:** in this step a very broad research area is chosen as the source of knowledge for the foundation of this research.
- **Step 2:** in this step we focus on retrieving and critically reviewing the existing literature in our chosen area of research. Also in this step, a survey-based approached has been used to investigate the state and characteristics of the current software development industry practice.
- **Step 3:** in this step the results of the survey and the literature review are clarified to help define specific questions for this research. As research questions have become clearer in light of more literature reviews, so our focus on the research area has become more specific.
- **Step 4:** in this step we begin to construct theories and hypotheses to answer the research questions.

(**Phase 2 – Technique**)

This Phase is a design Phase;

- **Step 5:** the process model ESIM is designed and recommended.
- **Step 6**: LifeEvent ontology, which is an essential knowledgebase for the practical deployment and implementation of ESIM, is designed in this step.
- **Step 7**: in this step of the research an implementation platform (LOOIS) is designed as a delivery platform for e-government integrated services.

(Phase 3 – System)

- **Step 8:** in this step the LOOIS prototype system is developed based on the recommended techniques of Phase 2 of this research process.
- **Step 9**: in this step an experiment is conducted to validate the LifeEvent ontology, ESIM framework and the performance of LOOIS prototype.

1.7 Thesis Structure

This thesis is consists of nine chapters, described as follows:

Chapter 1: is an overview of the research. It is intended to help readers obtain a cohesive picture of the research conducted in this study. It consists of four main sections: the problem analysis and research questions in this study, the brief description of contributions of this study to the body of knowledge, the illustration of the overall structure of this thesis, and the list of authors' publications related to this study.

Chapter 2: gives a clear description of definitions, technologies, and concepts to develop an understanding of issues associated with the current state-of-the-art. It provides a summary of the current general state of research in the public service domain and the mechanics of service delivery in most relevant areas of concern to this research. It also discusses a number of proposed models for e-government transformation and integration, critical analysis of the current state of research in the area of architectural and implementation models of e-government service integration, and of current technologies, such as the use of ontology, as a knowledgebase and reasoning tool.

Chapter 3: and the subsequent Chapters 4, 5, 6 and 7 will collectively present the main contributions of this study. Chapter 3 presents a comprehensive model for e-government integration. This model provides important guidance for the design and implementation of the integration framework discussed in Chapters 4 and 5.

Chapter 4: establishes our argument about the deficiency of current popular software engineering methods in dealing with web services integration projects. It proposes additional tasks for the classical software engineering process to create a more efficient new process for service integration engineering.

Chapter 5: provides a detailed design and formal description for LifeEvent ontology. It discusses the definition of ontology and the service ontology by defining the concept of 'Ontology'. Chapter 5 explains and illustrates the overview and details of our conceptual design for the LifeEvent ontology. It provides the annotation used in our formal method to establish the main axioms and design rules for the LifeEvent upper ontology.

Chapter 6: proposes specifications of a repeatable framework for e-government service integration projects. This process model, called E-Service Integration Modelling (ESIM), is a repeatable process, that defines the required tasks and activities for providing effective management control over the web service integration process.

Chapter 7: discusses the detailed design and implementation of the LifeEvent Ontology Oriented Service Integration (LOOSI) platform, which is a novel software application to provide the functionalities required to support our web services integration modelling processes called ESIM framework. The LOOSI platform is a proof of concept for all the theoretical proposals of this research found in Chapters 3, 4, 5 and 6.

Chapter 8: presents an evaluation of the proposed LifeEvent ontology by measuring its growth in complexity in comparison with OWL-S ontology. An evaluation of the LOOSI platform as a proof of concept validates the design assumptions of the ESIM framework and SIE.

Chapter 9: discusses the conclusion of the research and possible areas for future research. The relationships of all chapters are shown in Figure 1.3.

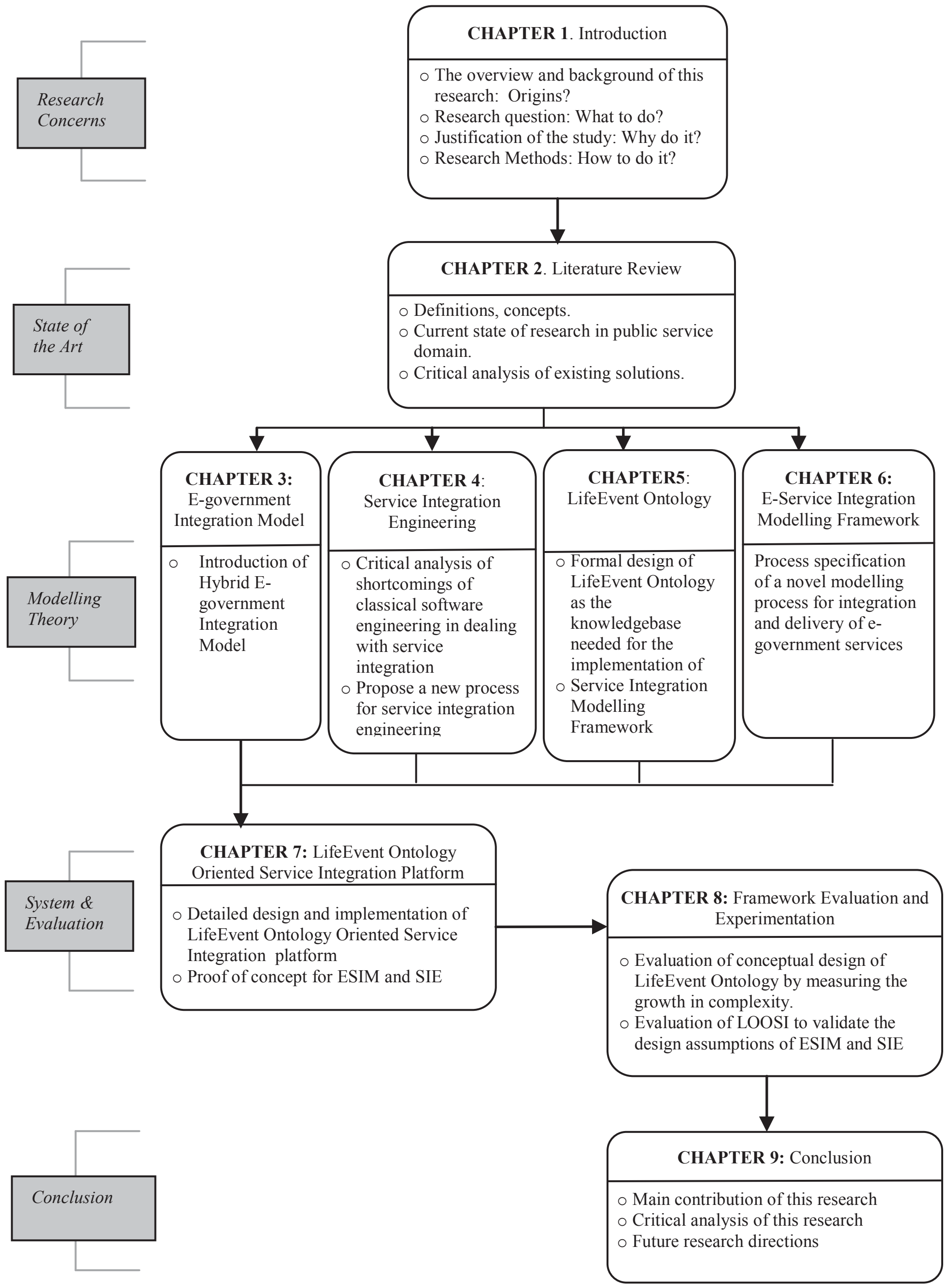

Research Concerns
State of the Art
Modelling Theory
System & Evaluation
Conclusion
CHAPTER 1. Introduction
The overview and background of this research: Origins?
Research question: What to do?
Justification of the study: Why do it?
Research Methods: How to do it?
CHAPTER 2. Literature Review
Definitions, concepts.
Current state of research in public service domain.
Critical analysis of existing solutions.
CHAPTER 3: E-government Integration Model
Introduction of Hybrid E-government Integration Model
CHAPTER 4: Service Integration Engineering
Critical analysis of shortcomings of classical software engineering in dealing with service integration
Propose a new process for service integration engineering
CHAPTER5: LifeEvent Ontology
Formal design of LifeEvent Ontology as the knowledgebase needed for the implementation of
Service Integration Modelling Framework
CHAPTER 6: E-Service Integration Modelling Framework
Process specification of a novel modelling process for integration and delivery of e-government services
CHAPTER 7: LifeEvent Ontology Oriented Service Integration Platform
Detailed design and implementation of LifeEvent Ontology Oriented Service Integration platform
Proof of concept for ESIM and SIE
CHAPTER 8: Framework Evaluation and Experimentation
Evaluation of conceptual design of LifeEvent Ontology by measuring the growth in complexity.
Evaluation of LOOSI to validate the design assumptions of ESIM and SIE
CHAPTER 9: Conclusion
Main contribution of this research
Critical analysis of this research
Future research directions

Figure 1.3: Relationship between chapters

1.8 Publications Related to the Thesis

The following are the list of my publications during my PhD study in descending date order:

1.8.1 Published Journal and Conference Papers

Sanati. F and Lu. J (2011), An Ontology For E-government Service Integration', International Journal of Computer Systems Science and Engineering (IJCSSE), ACCEPTED,2011.

Sanati. F and Lu. J (2010), "LifeEvent Ontology Oriented Service Integration," in IEEE International Conference on Service-Oriented Computing and Applications (SOCA 2010), Perth Australia, 2010, pp. 268-273.

Sanati. F and Lu. J (2009), "Life-event Modelling Framework For E-Government Integration," Electronic Government: An International Journal, 2009, vol. 1, pp. 183-202.

Sanati. F and Lu. J (2009), "Multilevel Life-event Abstraction Framework for E-government Service Integration," in 9th European Conference on e-Government London England, 2009, pp. 550-558.

Sanati. F and Lu. J (2008), "Semantic Web for E-Government Service Delivery Integration," in IEEE Information Technology Next Generation (ITNG) Las Vegas USA: IEEE, 2008, pp. 459-464.

Sanati. F and Lu. J (2007), "A Methodological Framework for E-government Service Delivery Integration," in eGovernment Interoperability Campus, Paris, http://80.14.185.155/egovinterop/egov07cd/eGov07-CDROM/pages /papers/ T1C.pdf, 2007, pp. 1-9.

1.8.2 Submitted Paper

Sanati. F and Lu. J (2011), ' Dynamic Integration of E-government Web Services: Framework and Implementation', International Journal of Decision Support systems.

Chapter 2: Literature Review

The primary aim of this study is to develop an intelligent integration model for e-government services to facilitate a repeatable and more efficient mechanism for delivering e-government services to customers. In this chapter, definitions, technologies, concepts and different integration models will be reviewed to develop an understanding of the issues associated with the current state-of-the-art. Section 2.1 will give a brief appraisal of the current general state of research in the public service domain. We will dig deeper into various areas of e-government research by identifying and explaining the main application areas of that research in Section 2.2. Section 2.3 is dedicated to discussing a number of proposed models for e-government transformation and integration. Sections 2.4 and 2.5 discuss the current state of research in the area of architectural and implementation models of e-government integration. Finally, Section 2.6 analyses current technologies such as the use of ontology as knowledgebase and reasoning tools as well as some best practices in the field of integration of web services in general. The overall aim of this chapter is to provide detailed knowledge about the state of the research and narrow the focus to e-government service integration and its delivery domain. The objective is to highlight the shortcomings and gaps in our knowledge in this domain.

2.1 E-government

In recent years, there has been rapid growth in the volume of research output on the topic of e-government. Researchers in the area of e-government seem to be working at the cross-roads of number of other research domains, particularly information systems, computer science, public science, and political administration, as are researchers in other fields including statistics, development studies, and linguistics. Research has been conducted on the philosophical analysis of e-government (Heeks and Bailur, 2007), but the literature comes predominantly from information systems. In particular, some of the e-government literature cited (Beer et al., 2006, Meneklis et al., 2005, Medjahed et al., 2003b, Peng et al., 2006, Irani et al., 2006) is essentially an IS or e-business idea that has somehow been modified to fit the public sector context.

By looking at the mechanics of e-service delivery by governments, one realises that a citizen-centric e-government service delivery platform must be built on the needs of citizens and government business partners if government is to develop an efficient and cost effective way to satisfy service user requirements. To achieve this objective, government has had to recognise the differences in the nature of its relationships with various customers based on primary needs and objectives. Three such government service relationships are recognised in the literature (Soliman, 2003) namely, G2C, G2B, and Government to Government (G2G). If we take one step further to include a technical vantage point, some literature has also described e-government implementation technologies in fine detail. Hepp (2006) and McIlraith (2001) very successfully predicted the technological trend of the e-government domain; however they have not gone so far as to propose a comprehensive and repeatable solution to the integration problem. Hence we see the need to dig deeper in to the roots of e-government integration and evaluate some well known e-government evolutionary models to identify the actual problems. Section 2.3 is dedicated to this purpose, but first we consider some application areas of e-government to get a sense of where these problems may have originated.

2.2 Key E-government Application Areas

To plan for e-government is to understand the key areas to which e-government is mainly applied. This section, though not comprehensive, discusses a number of application areas that are identified in the e-government domain. We only focus on the 'e-government transformation', which is mostly within the scope of this study. Nonetheless all the other areas need to be briefly explained for their importance and their probable influence on other areas of e-government research.

2.2.1 E-government Interactive Services

This constitutes the development efforts that are implemented to provide a broad range of services and a high level of functionality. Interactive service initiatives focus on providing information and content that is transactional in nature and offers a higher level of interaction. Communication capability is performed in complex mode (both citizens and government have the capability to provide feedback to one another).

One of the most desirable areas for the implementation of interactive services is in mobile government, also known as m-government.

A case study by (Chiu et al., 2007) extends an e-government appointment service into a mobile and ubiquitous one. In this case study, two main reasons are examined that highlight the importance of context in an interactive service. First, context reduces the input cost and improves efficiency. Second, context may provide an exciting user experience without much effort on the user's part. With the help of context, interactive services such as appointment reminders are accessories that can make citizens' lives easier by taking into account not just their location and environment (such as traffic conditions and weather), but also their preferences.

2.2.2 E-democracy and E-participation

Citizens are informed and educated about the decision-making process and programs to increase the transparency and responsiveness of government as a whole. Citizen feedback and involvement in day-to-day governance is encouraged as much as possible. Additionally, access to legislation and the decision-making process is more widespread, and efforts seek to bring government to the population and to as wide an audience as possible.

Research has highlighted the re-design efforts of e-government platforms offering public services. Some useful data supports this re-design. Citizens feel more confident and familiar transacting with local town-hall agencies. One survey has been conducted (Anthopoulos et al., 2007) for the local administration to assess the expectations of the local community with regard to e-government and make optimal choices for digital transformation.

In G2G services, on the other hand, most of the public agencies (87 percent) agree to the use of ICT systems to automate their routine transactions, according to Anthopoulos (2007).

E-democracy can also help improve citizen involvement and participation in decision-making and rule-making. Examples of e-democracy include the 'Lobbyist-in-a-Box' in the Commonwealth of Virginia where citizens can track the progress of bills as they move through the legislative process (Sheng and Trimi, 2008). The bill status information is updated every hour and users are notified of bill changes. Recent trials of e-government have met with public acceptance and eagerness. Citizens participate in online discussions of political issues with increasing frequency and young people, who traditionally display minimal interest in government affairs, are now the main group of participants.

Although Internet-based governmental programs have been criticised for lacking reliable privacy policies, studies have shown that people value the prosecution of offenders over personal confidentiality (Saumya, 2002). In the United States, 90 percent of adults approve of Internet tracking of criminals, and 57 percent are willing to forgo some of their personal internet privacy if it leads to the prosecution of criminals or terrorists.

Surveys and reports show a promising increase in public participation in e-government initiatives in general, regardless of minor setbacks that may arise from the fact that this is a relatively young area of research.

2.2.3 E-government Transformation

E-government is using Information Technology (such as software, hardware, computer networks, the Internet, and mobile computing) to improve government agencies' ability to transform relations with citizens, businesses, and other arms of government (The-World-Bank, 2007). The rapid growth in ICT has substantially dominated ordinary citizen's everyday life. We can recognise one visible extension of this change by the recent surge in public administration service delivery through the Internet domain. In recent years, many Australian government agencies (which are the main reference model of this study) have developed extensive e-government services for individual citizens.

This rapid growth in e-government services has resulted in duplicate and redundant services that could eventually lead to confusion for customers and added complexity in doing business with government where more than one government agency is involved. Customers who want to deal with government must first discover which part of government is providing their desired services, and this problem becomes more complicated if multiple agencies are involved. In this latter case, the problem of interoperability and incompatibility of these services will also add to the equation.

To prevent this issue from getting out of control, a number of proposals for e-government integration have been. Surveys of recent e-government integration solutions (Madhusudan, 2006, Umapathy and Purao, 2007) (Beer et al., 2006, Meneklis et al., 2005, Liu et al., 2007, Lu et al., 2004, Medjahed et al., 2003a, Peng et al., 2006, Dijkman and Dumas, 2004) indicate that most efforts have mainly relied on enabling technologies in order to achieve the desired outcome with very little attention being given to a unifying, methodological approach. Considerable research works have been done to design e-government implementation frameworks, some of which (Chircu, 2008) cover multidimensional aspects of e-government development, while others pay more attention to planning e-government development mainly from a project and resource management viewpoint (Ghapanchi et al., 2008). As far as this study is concerned, this fact is an indication of inadequate attention being given to define repeatable processes for e-government development.

2.3 E-government Evolutionary Models

In its 2009 report to European Union (EU) commission (Kotsiopoulos and Rentzepopoulos, 2009), the DG Information Society and Media indicated that integration attempts so far mostly belong to the so-called structural or syntactic level integration. This assumes relatively well-structured data, which allows rather tightly-coupled solutions leading to a single global schema. The EU commission report touches upon everything from data integration to organisational collaboration to semantic integration through the use of multiple ontologies.

2.3.1 Ziegler and Dittrich data integration model

The EU commission report has been built upon a research paper that originally proposed the classification schema presented by Ziegler and Dittrich (2004), which is based on a layered architecture for information systems.

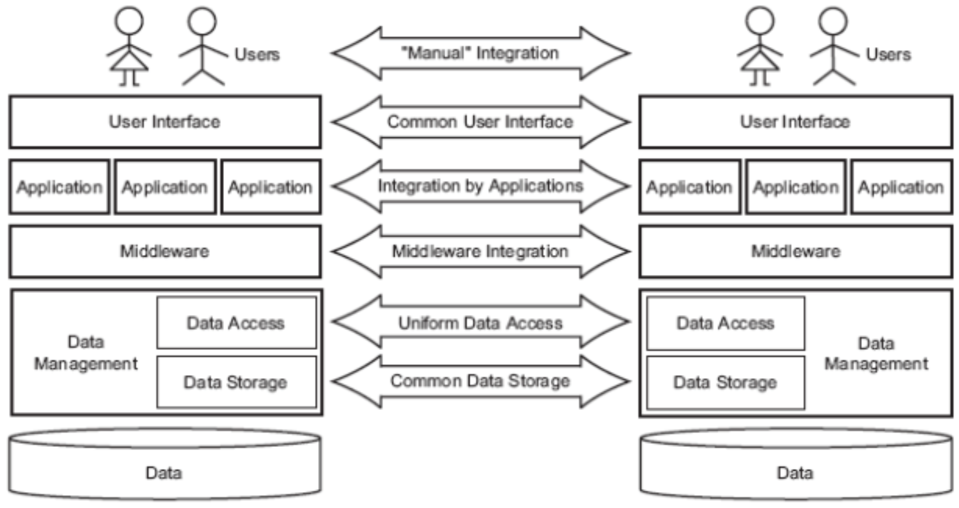


Figure 2.1: Ziegler and Dittrich (2004) integration model

Looking into the detail of the Ziegler and Dittrich integration model illustrated in Figure 2.1 reveals six layers in the e-government structure that could be exposed for integration.

1. **Manual Integration:** Users interact directly with all relevant information systems and manually integrate selected data.
2. **Common User Interface:** The next step towards more automated integration is a common user interface, such as a web browser.
3. **Integration by Applications:** This is the next level towards automated integration. Integration applications access various data sources and return integrated results to the user.
4. **Integration by Middleware:** Middleware provides reusable functionality that can generally be used to solve dedicated aspects of the integration problem (SQL-middleware for example).
5. **Uniform Data Access:** Logical integration of data is accomplished at the data access level. Global applications are provided with a

unified global view of physically distributed data, though only virtual data is available at this level.

6. **Common Data Storage:** Physical data integration is performed by transferring data to new data storage; local sources can either be retired or remain operational.

Other e-government integration models have been proposed that have certainly influenced the evolutionary trend of e-government. Amongst them we explain two models, one proposed by the Gartner organisation (Baum and DiMaio, 2000) and the other by Layne and Lee(2001).

2.3.2 Gartner E-government Model

According to Gartner's e-government evolution model (Baum and DiMaio, 2000), e-government has been evolving in four phases:

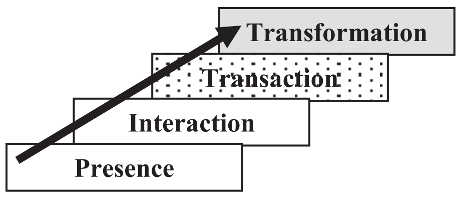


Figure 2.2: Gartner Integration Model (Baum and DiMaio, 2000)

- **Phase 1 – Presence:** in the first phase, governments are incited to create a homepage on the web.
- **Phase 2 – Interaction:** this phase provides the ability to download forms, perform simple searches, and e-mail government officials.
- **Phase 3 – Transaction:** in this phase, applications typically include functions such as an online transaction facility for constituents to pay parking fines, file tax returns, renew a driver's licence, and apply for building permits.

- **Phase 4 – Transformation:** In this final phase, governments start looking at creating web portals to serve as one-stop-shops for their constituents.

2.3.3 Layne and Lee E-government model

In another model by Layne and Lee (Layne and Lee, 2001), four stages of e-government growth are described as being:

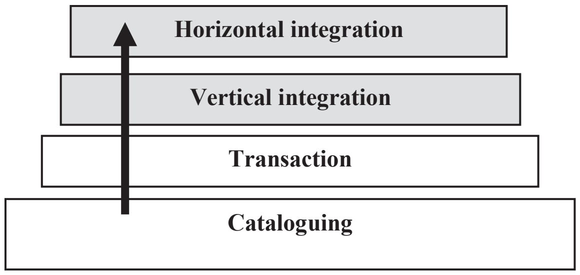


Figure 2.3: Layne and Lee (2001)Integration model

The Layne and Lee model is explained in terms of complexity involved and different levels of integration. In Layne and Lee's model, the first stage of e-government is called 'cataloguing' and focuses on cataloguing government information and presenting it online; for example, establishing an online presence, presenting a catalogue, and offering downloadable forms. The second stage is called 'transaction', which focuses on providing functions to citizens for transacting with government electronically. The functions include offering services and forms online, and working on database to support online transactions.

Citizens view government as an integrated information base by allowing the participation of agencies across different levels of governments and by having different agencies with different functionalities talk to each other. This model defines the third stage of e-government as vertical integration, which refers to local, state and federal governments being connected for different functions or services of government, such as when local systems link to higher level systems. In contrast, the fourth stage of e-government, horizontal integration, is defined as integration across different functions and services.

The Australian Government, as one of the leaders in public sector ICT planning and development in the world, has identified the urgency of integrating stove-pipe e-government services, but it is yet to formulate an interoperability framework for individual agencies, six year into the introduction of the plan (Reding, 2006). A number of research papers presented at conferences and in journals around the world are also dedicated to identifying the requirements of government transformation (Baglietto et al., 2005, Burk, 2005, Castellano, 2005, Evans and Yen, 2005, Fang, 2002, Grant and Chau, 2005, Heeks and Bailur, 2007). Recent publications (Gil-Garcia and Ignacio, 2007) suggest a modified model of e-government evolution, splitting the integration stage into three distinct but closely related sub-stages of Vertical, Horizontal and total integrations.

2.4 Web Services: State of Technology

Service Oriented Architecture (Zimmermann et al., 2004), and in a broader sense, Service Oriented Computing (Turner et al., 2003) have influenced ICT towards a design of uncoupled yet coherent architectures of services. Current ICT industry trends indicate that organisations and solution vendors are moving towards the decomposition of legacy complex processes into atomic and simpler components to handle the ever-increasing complexity of current information systems (Huhns and Singh, 2005). This trend has led to a two-phase solution: Phase 1 is to transform massive architectures into constructs, consisting of simpler building blocks, called services; and Phase 2 is to recompose these services into complex services in order to achieve added value. This research is mainly concerned with the second phase of this theory.

2.4.1 Software as a Service

Web services interact with one another dynamically and use Internet standard technologies, making it possible to build bridges between systems that otherwise would require extensive development efforts. Traditional application design depends upon a tight interconnection of all subordinate elements, often running in the same process. The complexity of these connections requires that developers comprehensively understand and have control over both ends of the connection; moreover, once established, it is extremely difficult to extract one element and replace it with another. By contrast with tight coupling principles that require agreement and shared context between communicating systems as well as sensitivity to change, loose coupling requires a much simpler level of coordination and allows for more flexible reconfiguration (Papazoglou, 2008).

For SOA to work, it is not enough to build and deploy a collection of systems and services. Services are meant to be shared, which means they must be created according to certain rules that everyone can follow. The collations of related rules are known as ‘standards’, without which an SOA cannot function.

Web services are also defined by an interface that can be formally stated using the Web Service Description Language (WSDL) (Chinnici et al., 2007), which defines and advertises the functions and behaviours provided by the Web service (Gudgin et al., 2006).

2.4.2 IT Industry Survey

One of the important areas that this research needed to understand was the readiness (knowledge and experience) of the Information Technology industry, particularly in Australia, for participating in e-government service integration. A recent online industry survey by DeSI laboratory (Sanati, 2010) at the University of Technology, Sydney (UTS) has gathered and analysed information from practising software engineers, project managers and developers to determine the common practices and tools used in the performance of their jobs. This survey was designed to gather data from current industry practices, in order to evaluate trends within the industry compared with the academic literature in the same area of knowledge. This survey was particularly designed to discover the tools and methodologies used in web services development and composition. It determines how these methods differ from Object Oriented Development (OOD) methods, and although the results of are not conclusive, nevertheless the information obtained in this survey can be used to direct further research into development and fine-tune the conclusions of this research.

A total of 40 survey participants were selected from entirely different industries, areas of work and responsibilities within the ICT industry. These organisations were:

1) Department of Education - State of New South Wales, Australia
2) Department of Justice, State of New South Wales, Australia
3) Centrelink - Australian Federal Government body

4) KAZ Technology Services - IT development and infrastructure, Australia
5) OPTUS - National telecommunication carrier, Australia
6) Woolworths IT division - retail industry, Australia
7) AAS - superannuation industry, Australia
8) Genworth Financials - insurance industry, Australia
9) ING Australia IT division – investment and banking, Australia

According to the statistics obtained in this survey, it appears that most of the ICT industry in Australia is lagging far behind in most areas of ideal software engineering practices, and specifically in web services integration engineering. The statistical results of the survey point to a higher degree of methodological uncertainty. Most companies tend to adopted experimental evaluation of different methodologies in order to find the most suitable one for their needs as a result of this uncertainty, even though in most cases such experiments prove to be a very costly practice. Following are some statistical facts from the survey to clarify our argument:

- Close to 35% of participants believe their organisation is not following any specific formal methodology in software development.
- One out of 10 developers says they have no or very little documentation on their development practice.
- Although 57% of IT managers believed their organisation is developing web services, only 4% are familiar with any formal Service Oriented Architecture or design patterns.
- Only 27% of the people who are developing web services are conducting an interoperability analysis for service composition.

A brief analysis of the aforementioned industry survey facts, specifically the last two points, in conjunction with the cited literature are evidence of the lack of industry adoption of recent results in the area of web services integration. This may be even worse in the e-government domain. We specifically stress the need for more research on modelling frameworks for e-government service integration and interoperability analysis.

2.5 Service Delivery in E-government

The report 'Australia's E-government Strategy – New Service Agenda' (Reding, 2006) indicates that the Australian Government has taken many serious steps towards planning and implementing the transformation and integration of its e-services across all agencies. Some planned developments in that report were overdue at the time of our review (March 2009) and at the time of writing, some have yet to be realised. Some other countries or community of countries such as the European Union have taken many more steps towards integrating their e-government services within member states. However, by taking a closer look, one can identify the main problem with some integration proposals as being not their technical solution but their conceptual perception of e-government (Mugellini, 2005). Although very expensive, it may well be possible to integrate all e-government services from all jurisdictions, in limited number of specific cases, from a technical perspective. We argue that the problem with this approach lies with the fundamental rule of separation of concerns between government agencies, known as what we would call a 'horizontal separation of concerns', and the limitation of jurisdictions at different levels, which we call the 'vertical separation of jurisdictions' (for example, Local, State and Federal jurisdictions in a federated system of government (Jaeger, 2002)). These aforementioned issues show that it is all but impossible to integrate and deliver e-government services from multiple agencies without a *common delivery platform*, both at the conceptual and physical level, that can consolidate all the jurisdictional, geographical and hierarchical differences. Later in this research we will propose such a framework and its delivery platform.

2.6 Semantic Web and E-government Interoperability

As we mentioned before, interoperability analysis is the key to e-government service integration. This section is dedicated to the investigation of:

1) the current state of technology in relation to general utilisation of semantic web, and
2) the definition, description and current use of ontology as a form of knowledgebase for semantic web.

The remainder of the Section 2.6 discusses the use of ontology in e-government and investigates the literature that introduced the concept of a 'life event' in the context of e-government.

2.6.1 Semantic Web

One of the most common barriers to the interoperability of systems, known as *Semantic Interoperability*, has proven to be much harder to overcome, according to a survey conducted in five European countries and the United States. According to Luis Guijarro's (2007) results, despite the surveyed countries' efforts towards achieving interoperability, Research has found that each of the six government agencies in the study has developed their own set of standards to address the interoperability. According to the survey the only common feature of the six interoperability frameworks is that Internet technologies comprise their core.

According to its 2006 report, the Australian government has identified the urgency of integrating e-government services (Reding, 2006), however it is yet to formulate an interoperability framework for individual agencies.

The semantic web is considered an extension of the World Wide Web, where information has well defined meaning, and it could enable computers and people to work in greater cooperation (Berners-Lee et al., 2001). The semantic web has become the default technological standard solution for achieving interoperability, particularly in the area of web services. It addresses semantics through standardised connections to related information (Helbeler et al., 2009). This includes labelling data in a unique and addressable manner.

The picture of a layered stack illustrated in Figure 2.4 outlines how the elements of the semantic web are related to form an architecture that supports semantic interoperability. The foundation of this stack is constructed by Unicode 3 and Universal Resource Identifier (URI). Unicode provides a unique number for every character, no matter what the platform, the program, or the language is (Inc., 2010), whereas URI is a compact sequence of characters that identifies an abstract or physical resource that underpins the semantic web by allowing machines to understand and process independently (McGuinness, 2004).

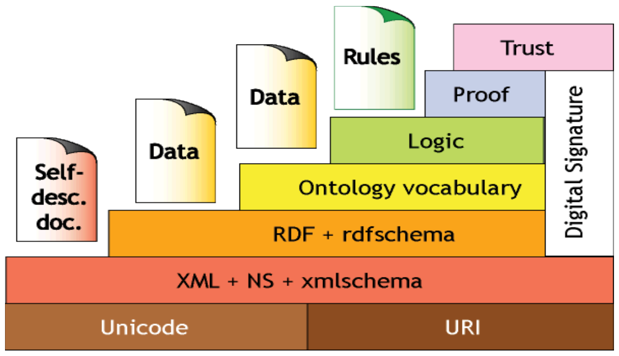


Figure 2.4: Layered architecture of semantic web (Berners-Lee, 2000)

On top of the first layer, Extensible Markup Language (XML), Name Space (NS) and XML Schema are built. XML standard is a simple, very flexible text format, designed originally to meet the challenges of large-scale electronic publishing. It is also playing an increasingly important role in the exchange of a wide variety of data on the web.

The flexibility of a web form enables connections to all the necessary information, including logic rules, and all these pathways and terms form a domain vocabulary or *Ontology*. Unlike relational databases that depend on a schema for structure, a knowledgebase depends on ontology statements to establish structure. Relational databases depend on one kind of relationship (foreign key) whereas the semantic web offers multidimensional relationships such as 'inheritance', 'part of', 'associated with' and many other types, including logical relationships and constraints.

Intelligent service delivery systems will require the use of an *Open World* (Yu, 2007, Helbeler et al., 2009) knowledgebase such as *Ontology* as opposed to relational databases that are considered *Close World* environments. Section 2.7.2 is dedicated to investigating the specifications of ontology as an open world knowledgebase.

2.6.2 E-Government and Life Event Initiative

Life Event is a metaphor to group more than two public services according to citizens' requirements. This group of services must be arranged to solve a problem or a need that relates to an event in citizens' everyday life (i.e. applying for a motorcycle licence or changing address).

A large number of e-government initiatives in the literature were scanned by this research to establish an understanding of the true state of e-government and specifically, life event initiatives among developed countries (Sanati and Lu, 2009a). This literature survey indicates that most of the efforts have relied on enabling technologies in order to achieve the desired outcome with very little or in some cases no attention to any methodological approach. Considerable work is being done to design e-government implementation frameworks, some of which (Chircu, 2008) cover multidimensional aspects of e-government development.

From the point of view of governments, the results are less exciting. Our scan of e-government initiatives in the area of service integration and life events shows that out of eleven independent states and countries within the developed world, none has actually implemented or initiated an e-service integration project and some have reduced the concept of a life event to simply providing information and directions about some government services (in some cases not even electronic services). For example, the Government of Ontario in Canada provides a page on its web site listing a number of 'life events', yet it only provides information about the physical service by the relevant department. This was similar in a number of states in Australia (Victoria, NSW, South Australia and Tasmania) (OntarioGov, 2007, AusTasmaniaGov, 2008, UKSalfordGov, 2007, Retirement-Investments, 2007, AusVictoriaGov, 2006).

Below, we discuss other e-government projects that have taken practical steps to deliver e-government services in a more organised and cohesive manner.

HELP is a comprehensive citizen portal of the Federal Chancellery in Austria and is considered as a One-stop shopping portal that is gradually becoming the norm for citizen services (AustriaGov, 2009).

In some jurisdictions, other initiatives based on life event orientation are often offered on the portal, bringing together relevant information for citizens related to a specific stage in life. For example, the Irish General Register Office life events site (IrishGov, 2010) provides automatic processing of child benefits claims. CAT365 in Spain (SpainGov, 2008) addresses education, training, and finding a job, and is mostly known as an integrated business creation service.

Such citizen portals are becoming more sophisticated by adding electronic identification, electronic payments and increased interactivity. The Finnish Centre for Pensions offers a web site (FinnishGov, 2008) in three languages on pensions, including a service to identified insured persons who use a personal authentication card to access internet banking. This service currently reaches some 80% of the working population. Although such initiatives are a great leap forward towards delivering e-government services in a better and more efficient way, they are barely touching the surface of true e-government integration.

Some good practices for truly integrated services do exist, according to the European digitizing public service ‘i2010’ (Capgemini et al., 2010). These examples offer a single entry point, i.e. a dedicated portal for the job seeker, guidance for the unemployed, and a focus on the desired outcome rather than simply fulfilling an agency's legal obligations. Figure 2.5 is a statistical graph illustrating the maturity of life events in different counties.

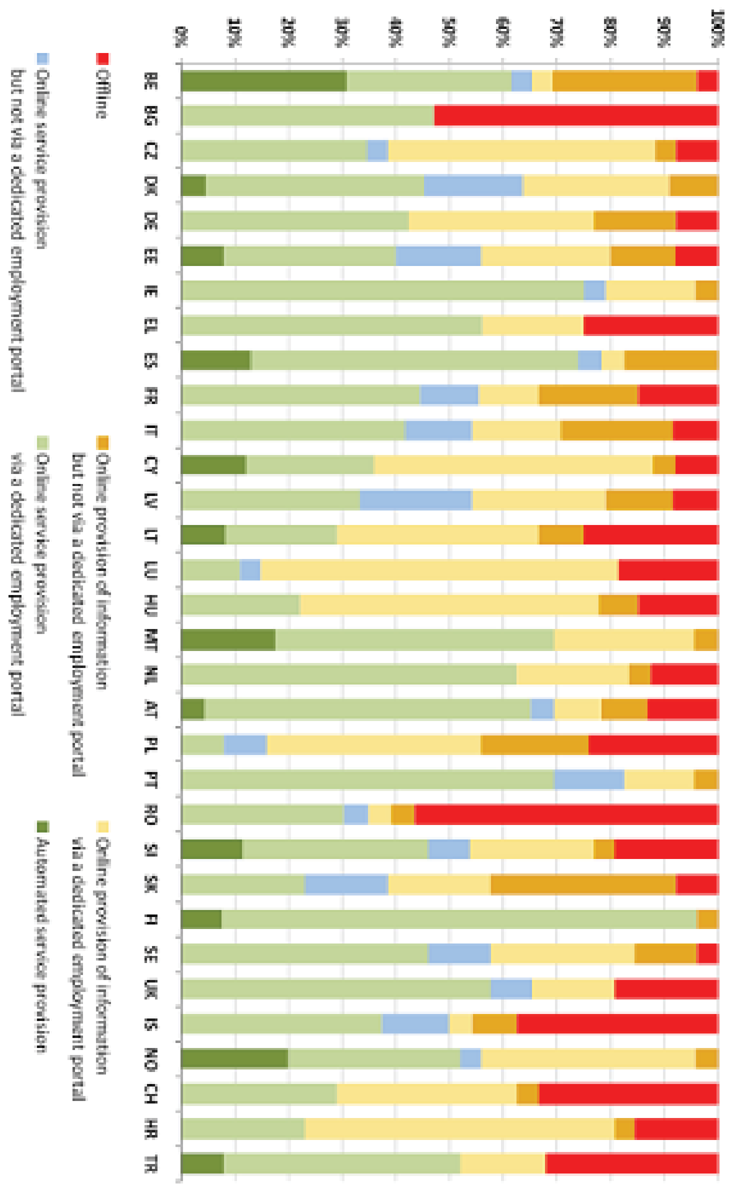


Figure 2.5: Maturity of the Life Event 'Losing and Finding a Job' (Capgemini et al., 2010).

2.6.3 Ontology for Semantic Web

Ontology is a term that was originally borrowed from philosophy by computing science. Ontology in philosophical terms can be seen as the study of the organisation and the nature of the world independently of the form of our knowledge about it. Thomas R. Gruber (Gruber, 1993) defines it as a specification of a vocabulary that describe a shared domain of discourse; in other words, an ontology is a formal representation of the knowledge by a set of concepts within a domain and the relationships between those concepts. It is used to reason about the properties of that domain, and may also be used to describe the domain.

For ontology to trespass into a physical presence and be used as the knowledgebase of a semantic web system, it needs to be structured around an expressive language. The Web Ontology Language (OWL) specification (McGuinness, 2004) is the recommendation of the World Wide Web Consortium (W3C). This ontology language has become more and more popular compared to other ontology languages; however, this research does not intend to explain all the predecessors of OWL as they are of little relevance to this study.

When ontologies become widely used for rendering rich semantics to data, it is possible to develop rules and inference mechanisms to take advantage of them. Highly intelligent web applications and services can be developed which will finally turn the vision of the semantic web into a reality. A number of research works presented at conferences and in journals around the world have identified *Interoperability* as one of the critical requirements of modern e-government transformation (Baglietto et al., 2005, Burk, 2005, Castellano, 2005, Evans and Yen, 2005, Fang, 2002, Grant and Chau, 2005, Guo and Lu, 2007b, Heeks and Bailur, 2007). From the discussion, the important position of ontologies in the formation of the semantic web and the solution of the interoperability of systems is obvious.

Ontology typically consists of a hierarchical description of important concepts in a domain, along with descriptions of the properties of each concept. Formally, ontology contains a set of *concepts* which constitutes the core of the ontology. The notion of concept in ontologies is similar to the notion of class in object-oriented programming. Each concept has a set of *properties* associated with it. This set describes the different features of the class. Each property has a *range* indicating a restriction on the values it can take. Ontology relates classes to each other through *ontology relationships*. Examples of relationships include 'subclassof' and 'superclassof'. Properties are also related through similar relationships such as 'subpropertyof' and 'suprepropertyof'.

2.6.4 Service Ontology

To make use of a web service, a software agent needs a platform-independent description of the service, and the means by which it is accessed. An important goal for any semantic web markup languages, then, is to establish a framework within which these descriptions are made and shared. Web sites should be able to employ a standard ontology, consisting of a set of basic classes and properties for declaring and describing services, and the ontology structuring mechanisms of OWL provide an appropriate representation language framework within which to do this.

A good starting point for understanding the web services paradigm is to consider the stated goals, as found in the literature and the standards communities. The basic motivation of standards such as SOAP and WSDL is to allow a high degree of flexibility in combining web services to create more complex ones, often in a dynamic fashion.

Diving deeper into the mechanics of web services description, composition and discovery in (Chinnici et al., 2007), Semantic Annotations for WSDL (SAWSDL) suggests how to add semantic annotations to various parts of a WSDL document, such as input and output message structures, interfaces, and operations. This extension is in line with the WSDL extensibility framework. SAWSDL defines a new name-space called 'SAWSDL' and adds an extension attribute called 'modelReference' so that relationships between WSDL components and concepts in another business semantic model (i.e. ontology) are handled. However SAWSDL 'modelReference' can only annotate concepts in the WSDL Metamodel and this makes it more difficult to implement an automatic service discovery and composition based on concepts that do not belong to the Metamodel of WSDL.

The OWL-S approach proposes an upper ontology for web services motivated by the need to provide three essential types of knowledge about a web service:

1) What does the service provide for prospective clients?
2) How it is used?
3) How does one interact with it?

The answer to Question 1 is given in the 'Profile' which is used to advertise the service. The service profile elements include: preconditions, inputs, outputs, results and service category. The answer to Question 2 is given in the 'Process Model' which includes: inputs, outputs, preconditions, effects and the behaviour of the service (data and control flow). The answer to Question 3 is given in the ‘Grounding’.

Web Service Modelling Ontology (WSMO) was developed (Domingue et al., 2004) as an ontology for semantic web services based on the Web Service Modelling Framework (WSMF) (Zhang et al., 2004). WSMO defines four top level elements as the main concepts which have to be described in order to describe semantic web services, namely ontologies, services, mediators and goals. A service is described in terms of non-functional properties, a provided interface and a provided capability.

Much more ambitious goals are adopted by the OWL-S working group (W3C, 2004). These goals seek to provide machine-readable descriptions of web services which will enable automated discovery, negotiation, composition, enactment, and monitoring of web services. OWL-S is an ontology language for describing web services in terms of their inputs, outputs, preconditions and effects, and of their process model. Importantly, OWL-S provides a formal mechanism for modelling the notion of the state of the ‘real world’ and describing how atomic web services impact that state over time. OWL-S also provides a grounding, which provides mechanisms for mapping an OWL-S specification into a WSDL specification. A kind of middle ground is also emerging, which provides abstract ‘signatures’ of web services that are richer than WSDL but retain a declarative flavour.

The closest recent research to our work that we encountered was on web services resource adoption (Forte et al., 2008) that proposes the use of ontologies and web services within a framework of components for the content adaptation domain, to facilitate the development of software based on reuse of existing web services. OWL-S with its current specification provides adequate facilities to enable a knowledgebase web services discovery and invocation. However, what is missing is:

1) support for composition of web services from multiple web services providers and multiple web service WSDL.
2) infrastructure and mechanics to support the intelligent construction of complex integrated e-government web service with the capability of enforcing government regulations at runtime;
3) construction of meta-workflows that could be instantiated with alternative services based on user requirements;
4) Dynamic re-construction of complex composite web services at runtime in the event of individual service failure.

2.6.5 E-government Ontology

There are a number of works in the area of e-government ontology, two of which we found to be closely related to our work in this research. The first, *The Ontology-enabled e-Government Service Configuration Project* (ONTOGOV, 2006)', constructs services ontology via a domain expert using a tool called Service Modeller. The domain expert also adds more semantics by creating instances of the following ontology used throughout the project. This project has classified and uses a number of separate ontologies as the knowledgebase of their e-government system implementation.

- **Domain ontology**, comprising concepts like data (e.g. name, first_name, municipality_from, municipality_to) and documents (e.g. application form, administration leaflet).

- **Legal ontology**, comprising instances of process relevant law or regulations, e.g. the basis of the new process is a regulation about settlement. Several instances will then be initiated in the legal ontology indicating the related law, paragraph and article.
- **Organisational ontology**, comprising instances of process relevant to organisational units, e.g. involved in the new service are the organisational units 'Registration Office' and 'Administration Office' with its roles and personnel.
- **Lifecycle ontology**, comprising instances of all design decisions relevant for the new service (e.g. technical or process immanent reasons), including instances of legal and organisational ontologies.

The second is an ontology project called *Impact of eGovernment on Territorial Government Services* (TERREGOV) (Anderson, 2006). In a similar fashion to the ONTOGOV project, it is also a multipurpose ontology that models the domain of public administration activities. It is used for semantic document retrieval, semantic discovery and the orchestration of web services, and so on. The same ontology is handled by multiple actors responsible for providing the technologies supporting these various functionalities.

From this project was developed 'Local Government Ontology' which enables local administrations to deal with information as a strategic resource. The ontology itself is reusable (a first example of data/knowledge reusability); it has been employed as the first step of the QUALEG project starting ontology, which was later expanded to fit specific project needs.

The aforementioned ontology projects were developed as part of two e-government initiatives developed by the EU five and six years ago respectively, but we found no evidence of further development or progress on those projects after the initial effort. One has to remember that a single ontology relating to public administration must provide multiple functionalities, and must allow its maintenance and editing by multiple users, including domain experts with no particular expertise in ontology engineering. This makes handling and maintenance extremely complex.

2.6.6 Service Ontology for E-government Services

Research publications as early as the start of this century have emphasised the role of ontology in e-government integration (Grosof et al., 2004, Lara et al., 2003). A study carried out by Stojanovic and Apostolou (Stojanovic et al., 2006b) was interested in the practical implications of ontology in e-government integration and analysed interoperability issues in the e-government domain. This work lists only a set of functional requirements for ontology building, and seems to overlook the overall qualitative criteria that ontology should be addressing.

2.6.7 Concept of Life Event Ontology

A widely accepted paradigm called the 'Life Event' model (Sabol and Mach, 2004, Skokan and Bednar, 2008, Stojanovic et al., 2006a) was introduced that can effectively improve the situation by supporting the e-government web services integration tasks in all their uniqueness and complexity. Early attempts to model LifeEvent Ontology (Trochidis et al., 2007) have mainly resulted in producing separate ontologies to provide a knowledgebase for the functional requirements of the systems that use them. There are also other research cases involving the concept of LifeEvent (Momotko et al., 2008) that do not consider ontology as a means of managing the services knowledgebase, but rather suggest modelling LifeEvents as workflows of related public services and actions. In other literature such as (Beri. and Vintar, 2004), the concept of LifeEvent is reduced to no more than a taxonomy of terms in a certain domain.

The importance of LifeEvent in e-government web service composition was recognised in recent research work by (Wolf and Krcmar, 2008). Other research work has identified the importance of requirements elicitation and critical factors in adopting e-government models (Shareef et al., 2009), although they mostly focus on one aspect of e-government. For example, the model presented by Wolf and Krcmar (2008) seems to be a very specific application only designed for B2G, and it suggests a model of features and phases that might not provide sufficient analysis for the further development of such a model. One work which is highly related to our research in e-government integration (Chiu et al., 2007) argues that, in new interaction devices, the context in which a service is being used becomes an integral part of the activity carried out with the system, but that research doesn't deal with question of how to persist the application context? We believe that LifeEvent ontology is the proper answer to this question.

Analysis of other relevant literature (Trochidis et al., 2007) recognises two main approaches for modelling lifeEvents. The first approach suggests modelling lifeEvents as workflows of related public services and actions (Trochidis et al., 2006). The second approach suggests modelling LifeEvents using ontology (Peristeras and Tarabanis, 2006) and thus capitalises on the idea of semantic representation of knowledge. This model describes ontology as the network of connections between concepts of a particular domain with the aim of providing a well-structured model. This research is built based on the design assumptions of the second approach.

2.7 Summary

In this chapter we reviewed a broad spectrum of literature related to all aspects of e-government service integration and delivery. There are four major areas of knowledge recognised as being of concern for evaluating the state of research in e-government.

The first area of knowledge is in the different approaches to the e-government evolutionary model, for which we argue that a more comprehensive model is needed to reflect more of the reality of the e-government evolutionary transformation trend.

The second area of knowledge is the type, the state and the maturity of the technology to support e-government integration. It seems that the majority of researchers agree that the web service and SOA are the most appropriate and popular.

The third area of knowledge is the mechanics of supporting an intelligent architecture to deliver dynamically assembled and configured composite web services. OWL-S and WSMO have had the most influence in this area of research out of many initiatives. We highlighted three major shortcomings of these technologies that need to be compensated for if a truly dynamic integration of e-government services is to be achieved.

The fourth area of knowledge is required to achieve a dynamic integration as a repeatable and comprehensive framework capable of delivering such a complex task. We believe that the concept of LifeEvent is well suited for use as the basis of such a model.

Chapter 3: Hybrid E-government Integration Model

This chapter and subsequent Chapters 4, 5, 6 and 7 will collectively present the main contributions of this study. This research presents in this chapter arguments regarding the most practically relevant and technically possible evolutionary pathway to e-government transformation, based on the body of evidence presented in Chapter 2 with the special reference to the review of recent literature in Section 2.3 that has the most relevance to the contributions of this chapter.

3.1 E-government Evolutionary Development

In recent years, empirical studies have identified two interesting dynamics in e-government evolution. First, e-government has evolved from its initial presence on the Internet to a more transactional and integrated approach. Second, as a general trend, national governments have started adding technological sophistication that has been perused by federal, state and local governments.

In the context of e-government evolutionary development, our theory that is detailed in this chapter, explains the future trend of e-government development at the service delivery level.

E-government has been acquiring more technological and organisational sophistication as a result of both institutional interoperations and pressures from different sectors of the society such as citizens, politicians, businesses, interest groups, and other stakeholders. In addition, e-government initiatives are evolving from the national to the local level. Our analysis of e-government evolution in Chapter 2 indicates that rapid growth in e-government services has resulted in duplicated and redundant services that could eventually lead to confusion of customers and add to complexity of doing business with government where more than one government agency is involved. Customers who want to deal with government must first discover which part of government is providing their desired services, and this problem can get even more complicated if there are multiple agencies involved, in which case the problem of interoperability and incompatibility of these services will also add to the equation.

To prevent the propagation of this issue, some proposals have been published regarding plans for e-government integration. The analysis of the past proposals have lead this research to propose a new integration model for e-government that better suits the current trend of e-government evolution.

Gou and Lu (2007a) have introduced a new e-government evolutionary model comprised of four stages. The model integrates Layne and Lee's (2001) model with Gartner's (2000) model in the context of Australian government online services. The four stages of this model are:

1) Government service online
2) Transaction-based government online services
3) Integrated government online services
4) Intelligent government online services.

The scope of the research conducted by Gou and Lu in the area of e-government development model did not extend to explain the practical implications and the model of implementation. However it provides a road map that better suites the e-government development trend in the real world. Therefore we use the Gou and Lu model as the foundation of its new proposed model for e-government evolutionary development. We take the Gou and Lu's model one step further to add more details specifically on the Stages three and four of the model. The model proposed by this research is a step forward to predict and explain the practical implications and the possible method of implementation for Stages 3 and 4 of the Gou and Lu model.

3.2 A Hybrid Model

The general understanding of the models discussed in Sections 2.3.1, 2.3.2 and 2.3.3 clearly identifies four possible categories of e-government integration models. It is the understanding of this research that most proposed e-government integration models can more or less fit into the following categories.

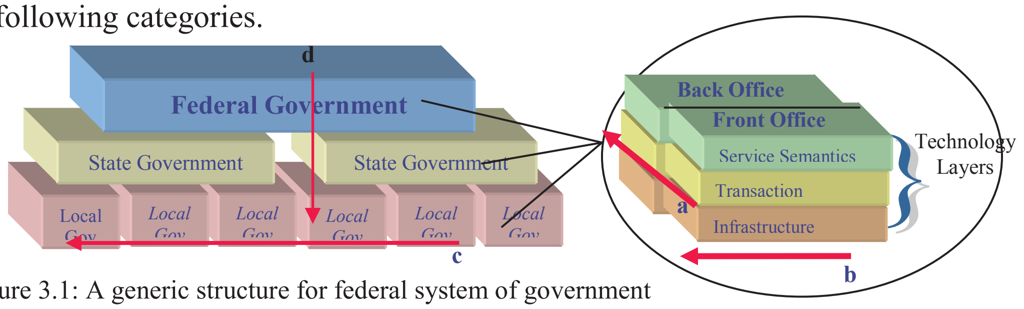


Figure 3.1: A generic structure for federal system of government

a) *In different depths*; to cover only front offices or across the border covering front-office and back-office (Chiu et al., 2007).

b) *In different implementation levels*; this could cover semantics or could go down to cover transaction and/or infrastructure sharing (Lü, 2007).

c) *Horizontal*; covering many government agencies at the same level (local or state or federal).

d) *Vertical*; cutting across different levels of government such as local, state and federal government agencies (Shahkooh and Abdollahi, 2007).

All integration models reviewed in this research either do not mention or are implicitly assume the integration of the back-office as part of their model. The integration of e-government back-office is not a trivial task. The first problem with back-office integration is that its information content and infrastructure protected by law to some extend (i.e. legislation such as the Privacy act). The second problem with integrating e-government at the back-office level is the logic surrounding the modern multilevel government structures. To illustrate this problem we argue that the difficulty with this approach lays with the fundamental rule of separation of concerns between government agencies (horizontal separation of concerns) as well as the separation of jurisdictions at different levels (vertical separation of jurisdictions, for example Local, State and Federal jurisdictions) around which government is structured (Jaeger, 2002). Based on this logic we can conclude that the integration of the e-government back-office is unlikely to produce any added value to the current state of e-government even if legislative and structural impasses imposed by the current protective laws are ignored.

The existing e-government models cannot meet the current e-government practices in its current state. Therefore a new evolutionary development model is needed in order to achieve better services that are designed to satisfy the ever increasing demand for more sophisticated and complex services.

Based on the observations of this research described in chapter 2 we have proposed a hybrid model that can explain the evolutionary transformation of e-government. This hybrid e-government integration model can achieve its goals, only if it is based on the real life requirements of citizens. In most cases e-government services must be integrated and packaged from both vertical and horizontal government jurisdictions to form a composite service. One simple example of this fact would be to apply for a ‘postgraduate scholarship’ from Australian federal government; clearly the integration of government services to fulfil this life event would require the collaboration of a number of state and federal agencies as well as number of private businesses such as the university itself.

In most cases complex services are needed to satisfy the life event requirements of citizens. There is no guaranty that the service components of such complex services would only come from one agency or even only from one level of government (Local, State or Federal). Therefore it is the position of this research that an innovative and comprehensive integration model is needed to facilitate the integration of government services at all levels.

Introduction of the Hybrid E-government Integration Model can overcome all the shortcomings of other integration models. This new model is a crucial building block within the complete solution for e-government service integration and delivery specified in this research. This model provides the much needed philosophical and organisational guidance for the design and implementation of this research.

3.3 Specifications of Hybrid E-government Integration Model

We mentioned that a more comprehensive model is needed to reflect on the reality of the e-government evolutionary transformation trend as it occurs in the real world; this model is described in Figure 3.2.

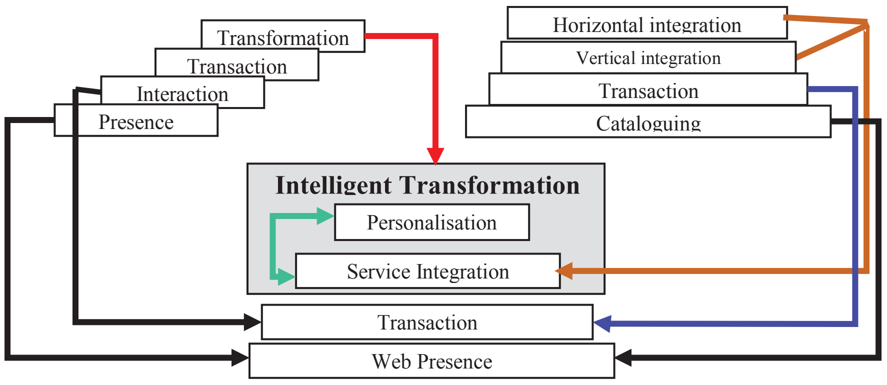


Figure 3.2: Hybrid E-government Integration Model

This new model is a combination of a number of models described in Chapter 2 with some modifications. It consists of three main levels:

3.3.1 Web Presence

This is an initial presence, when a government body has a formal presence on the Internet through a limited number of individual governmental pages (mostly developed by single governmental agencies). Governments in this stage normally offer static information about agencies and some of the services they provide to citizens and private organizations. In this level government agencies develop their own Internet information delivery systems to their customers, business and other government agencies.

Later this level evolves in to more dynamic presence, specialised information delivery system that is distributed and regularly updated in a large number of government sites. There are other cases where a national government official site serves as an entry point with links to pages of other branches of government, ministries, secretariats, departments, and sub-national administrative bodies (OntarioGov, 2007). Some governments might start using electronic mail or search engines to interact with citizens, businesses, and other stakeholders.

In this level governments generally use a state wide service or a national portal as the initial page providing access to services in multiple agencies. The interaction between citizens and different government agencies increases in this stage (e.g., e-mail, forums).

Citizens and businesses can access information according to their different interests. In some cases, passwords are used to access more customized and secure information.

3.3.2 Transaction

This level of improvement would allow people also use government websites to complete transactions. This phase of e-government would include a database link to online user interfaces. Portals are created as specialised information and transaction delivery channels. These portals usually become a unique showcase of all the governmental services available in the relevant area of interest. The needs of different constituencies are the main criteria for portal design and access.

These portals allow secure electronic payments to be made, facilitating transactions such as tax, fines, and services payments.

3.3.3 Intelligent Transformation

This level facilitates the integration and personalisation of government online services to enable users to cut across services provided by all government agencies in order to create their own personalised composite services. Cross agency integration encompasses the integration of logically and legally related web services provided by different departments of government. Horizontal integration between different governmental services must exist for citizens and other stakeholders to have access and take advantage of all the potential capabilities of information technologies in government. Therefore, in this stage governments need to cross the organisational boundaries and develop a comprehensive and integral vision of the government as a whole.

E-government services can also be available within the hierarchy of the same agency, which is called the ‘Vertical Integration’. Vertical integration encompasses the integration of similar services provided by different levels of government. Therefore, it does not refer solely to an initial integration in the form of government web sites but to the change and reconstruction of all their electronic services to become available as Web Services.

Intelligent Transformation refers to the situation in which government services are fully integrated (vertically and horizontally. Citizens have access to a variety of services through a single point of access. All services can be accessed from the same web page Citizens must be able to construct and personalise new composite web services from readily available e-government services. A transformation unnoticed by the public has taken place, and now services are organized according to citizen everyday life event requirements. An integrated online service delivery channel is also needed for the implementation of this level, which will link the existing delivery channels.

Intelligent Transformation of government online services provides high quality information in an integrated communication environment, which may evolve into a knowledge management system later in the future. Delivery of this phase will require online integrated services to become adaptive, personalised, and dynamically configurable. In this stage government transactional online presence will transform in to a more intelligent presentation of web content delivering more personalised and integrated services.

E-government services are aimed at the satisfaction of citizens life event needs. The relation between government administration and governed citizens should be described in terms of services. One can observe aspects of a provider–consumer relation just as in other services, but there is usually no market to choose a provider, and in many cases citizens may not choose at all since they are forced by law to use government services.

However, besides efficiency or other money saving goals, e-government plans frequently emphasise the improvement of service quality. Taking this seriously, we follow service as the guiding vision and try to explore what it needs to enable the service provider in order to fulfil their task. For example the satisfaction of citizens' needs based on caring for the citizens' concern. There is no question that the quantity and quality of e-government programs are rapidly increasing. They promise a means to deliver better service at a lowered cost. But no matter how brilliant the idea, it is worthless unless citizens adopt those services.

Given these requirements, an e-government integration model of web based self service points of access with the actual operation carried out somewhere in the back-office does not seem to be a suitable solution. Instead, as usability and flexibility throughout the process are success factors for relationship based services, this research introduces a framework for total integration and delivery of e-government services to satisfy goals intended for intelligent transformation of e-government. This framework discussed in Chapter 6 provides a centralised point of access to integrated e-government services.

3.4 Summary

This chapter introduced the Hybrid Government Integration Model. This model provides a conceptual guiding metaphor of e-government evolutionary development at its final stage of transformation. This chapter provides the philosophical foundation and guidance needed for the novel solution discussed in the next four chapters of this thesis that can enable the Intelligent Transformation of governments. The requirements are elicited based on observation and analysis of a number of existing models of e-government development in the literature. The Hybrid Government Integration Model discussed in this chapter plays an important role on the success of this research in designing a practical framework (ESIM) in Chapter 6 that can provide a comprehensive solution for e-government service integration and delivery.

Chapter 4:
Service Integration Engineering Process

This chapter is dedicated to establishing our argument about the deficiency of current popular Software Engineering (SE) methods in dealing with web services integration projects. We argue that current practices of SE that are mainly used for the development and maintenance of software applications are not efficiently capable of addressing most of the theoretical and technical issues raised by the emergence of web services integration projects. This chapter therefore focuses on introducing the concept of the Service Integration Engineering (SIE) process in the e-government domain by presenting a comparative analysis of the current state of SE practices and what we perceive to be the function of SIE. Before we commence our analysis of and into describing the design of SIE, we give a brief description and specifications of SIE to clarify our view of SIE in this. We will then present our comparative analysis of SE vs. SIE to establish why they differ from each other. To do this, we make a comparative analysis of SE and SIE in Section 4.1 and explain the concept of LifeEvent and LifeEvent abstraction in Section 4.2. In Section 4.3, we explain how the LifeEvent model effectively supports the e-government service integration task.

Later in this thesis we demonstrate how to arrive at the semantic interoperability of services in order to achieve e-government integration at the service delivery level. To ensure the repeatability of such a process we put together the unified modelling approach that is explained in Chapter 6.

4.1 Integration Engineering vs. Software Engineering

Although it is commonly perceived that Web Services are technology solutions designed to add agility to business processes, experience indicates that technology solutions rarely deliver agility unless they are primarily focused on business objectives. Technology vendors claim to add agility to the e-service development process, trying to sell integration and development tools which mostly focus on ‘Business Process Orchestration’ or ‘Web Services Workflow’ (Arroyo et al., 2006). They use a technology-driven approach that essentially misses the main point of the services. Arguably, as always with IT, the focus falls on technology rather than methodology.

Recent literature (Siew, 2006) has investigated and analysed existing component-based agile software methodology to identify the gaps for web services development. A comparison study of web services development and agile methodology has identified the additional steps required for web services development. A traditional component-based software development approach based on waterfall methodology consists of five distinguished phases, namely Requirements, Analysis, Design, Implementation and Testing. Although each phase is intended to be independent, there is always the possibility of iteration between the development phases to ensure the correct design and complete implementation of the requirements. Artefacts or outputs are produced at the end of each phase, which in turn is the input of another phase and also becomes the benchmark for a test phase at the same level.

Software Development Life Cycle (SDLC) models are designed to guide the development activity to correctly follow a series of steps in creating softwares to meet business needs. The SDLC models have evolved as new technology and new research has addressed the weaknesses of older models. Ideas have been borrowed and adapted among the various models.

Integrating existing e-government web services could be a challenging engineering task in that is different from developing web services from scratch. We compare the generic classical SE tasks including the proposed SOA extension (Siew, 2006) (Oracle, 2010) with SIE tasks and activities in order to represent the differences from a comparative perspective.

As illustrated in Table 4.1, every phase in a classical SE lifecycle has a corresponding phase in our proposed SIE lifecycle although they perform different tasks and activities to produce appropriate outcomes of each phase. Some interoperability analysis and provisions have become necessary SE tasks in recent years, mostly due to the distributed nature of modern businesses. This highlights the importance of essential interoperability requirements for distributed databases and applications, however, this interoperability analysis and design has never been an integral part of the software lifecycle and has only been performed in an informal manner as additional activities fitted in between other tasks in all phases of the SDLC.

	Software Engineering (SE) tasks	**Service Integration Engineering (SIE) tasks**
Analysis	Identify business requirements and translate them in to functional and non-functional requirements.	Specify the LifeEvent requirements of a generic citizen or a business partner.
	N/A	Specify inter-agency web service interoperability requirement at semantics and regulatory levels.
Design	Translate the requirements into conceptual models. Analyse requirements to define high-level design structure.	Identify all existing e-government services that could play a role in performing and completing a LifeEvent. Identify the service interface contracts.
	Design major infrastructure and system components (use design patterns).	Construct a Metamodel for the service group (composite service) from available Meta Services, required to perform and satisfy the LifeEvent.
	Describe the responsibilities of all system objects and their relationships for every component.	Define and Configure the relationships between all service component groups with service providers and service consumers (Implement and Configure regulatory requirements).
	N/A	Specify Ontology Instance Specifications for the LifeEvent.
Delivery	N/A	Interoperability requirements Implementation (inter-agency information and workflows)
	Implement all the system services, interfaces and business components as well as system and data configurations.	Populate LifeEvent ontology instances and intelligent algorithms.

Table 4.1: Comparing activities in SIE and SE

Service integration engineering is about configuring the interoperability of existing services to create new composite web services and workflows so that they can deliver to customers better and more intelligent services. More or less 50% of the anticipated tasks in SIE are interoperability-related, as is illustrated in Table 4.1.

Testability of every phase of the software life cycle is also a crucial component of software project management, and it is imperative to organise all technical, semantic and organisational interoperability tasks and activities into formal and testable phases of the SIE process to ensure a successful web service integration project.

Table 4.1 is a comparative analysis of our proposed SIE with the classical software engineering process described by Sommerville (2004) in his book *Software Engineering*. Analysis of this comparison indicates very limited similarities between the two, and the greatest emphasis in SIE is on solving complex interoperability problems and service workflow configuration rather than designing new components.

Our analysis indicates a substantial difference between activities performed in the SIE project and a typical SE that uses component-base software development methodology.

Interoperability specification and enabling tasks seem to have not been previously considered as an integration-specific stage in any SDLC, yet we believe these two steps are crucial to any successful e-service integration project. Important new concepts such as LifeEvent in the context of e-government services (Castellano, 2005), which are described as citizens' basic service requests, such as applying for a driver's licence, are also new to the classical software engineering. Newly-introduced concepts need to be more thoroughly analysed in a service integration context in general and in e-government integration specifically.

A comparative analysis of both processes detailed in Table 4.1 can be summarised as follows: LifeEvent is considered to be the fundamental unit of requirement for e-government service integration projects, as opposed to 'functionality' which is the simplest unit of requirements in the modern object oriented software development process. In a broader sense, from the citizen's point of view, it comes down to using one or more e-government services to satisfy a typical citizen's basic private or social need.

The simple fact that the complexity of e-government service integration is rapidly growing as a result of the constant increase in available services further highlights the need for a formal methodological approach and design standards to ensure efficient - and more importantly, repeatable - service integration process.

The substantial difference between the two processes provides enough reason to believe that the traditional software development frameworks are ill-equipped for efficiently dealing with service integration projects, not to mention the theoretical and technical issues discussed in Chapters 5, 6 and 7, which may increase the risk of complete or partial project failures. Therefore, we are encouraged to take this study one step further and research the requirements of a new framework that would be capable of carrying out successful and efficient service integration.

Repeatability is the most important key attribute of modern engineering frameworks. From a service integration point of view, it is important to identify services and their formats that are used by many SOA participants (agency e-services in this case) in order to drive our integration framework standards. This will ensure the reliability and scalability of the integrated services, as well as the repeatability.

4.2 LifeEvent

LifeEvent is a concept that we use as guiding metaphor for customer-centric public service provision. From an e-government integration point of view, a LifeEvent is a collection of actions including at least one public service, which when executed in its appropriate workflow fulfils the needs of a citizen arising from a new real-life situation (Trochidis et al., 2007).

This section explains the principles used for the analysis and modelling of LifeEvent. The concept of constructing a LifeEvent from available web services is illustrated in Figure 4.1.

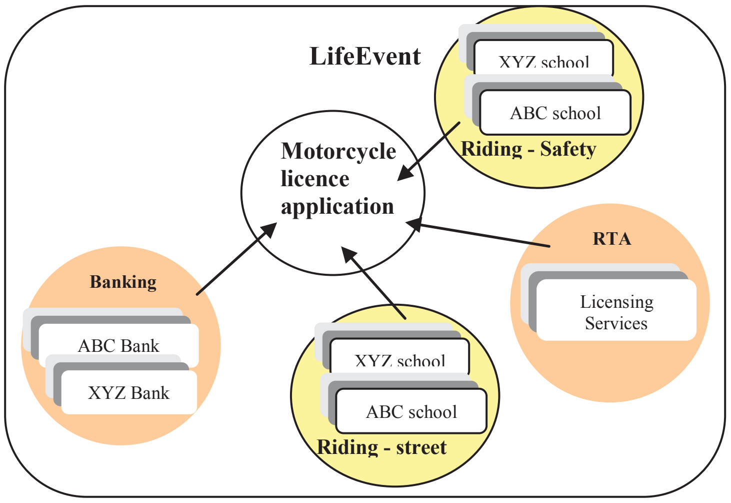


Figure 4.1: Constructing a LifeEvent from available web services

The concept of abstraction in an object-oriented paradigm (Kramer and Hazzan, 2006) plays an important role in the representation of complex data structures. Abstract objects or data structures can form hierarchical representations to provide easy-to-understand solutions for complex models. Abstraction is the means by which only a certain level of information detail is exposed by the entity, depending on the levels of representation intended for that model.

Different levels of data abstractions are also known as the level of granularity of a model. This study invokes the principle of data abstraction in the context of LifeEvent to represent an integrated service in different levels of granularity, depending on the detailed information about its underlying service structure and business rules.

The LifeEvent model effectively supports the e-government service integration task in all its uniqueness and complexity by combining basic services offered by multiple public authorities into a single composite service that corresponds to an event in a citizen’s life. We use the concept of LifeEvent to create the building blocks of an integrated e-government service delivery system. This concept is illustrated in Figure 4.2.

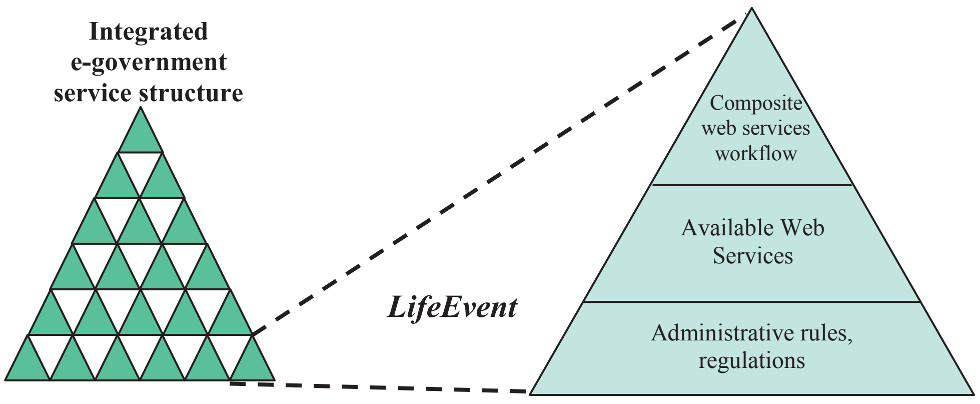

Integrated e-government service structure
LifeEvent
Composite web services workflow
Available Web Services
Administrative rules, regulations

Figure 4.2: LifeEvent as building blocks of integrated e-government

If it is properly modelled and implemented, LifeEvent has the capacity to revolutionise the way government web services are analysed, modelled, composed and delivered to provide a citizen-centric view of the government model. We consider the concept of LifeEvent to be almost equal to the concept of 'composite service' within the scope of this research, and we use the words *LifeEvent* and *composite service* interchangeably. Examples of LifeEvent are *Moving House*, *Passport Application* and *Motorcycle Licence*. Throughout this research, we will use the *Motorcycle Licence Application* example to demonstrate the related concepts.

4.3 Service Integration Engineering in Practice

This section is dedicated to discussing in detail the main activities of our proposed SIE process. It is important to note that the services integration specification is live and subject to change through feedback within the process. Based on the discussions in Section 4.2, this research suggests additional activities and techniques that are required to handle the complexity and unique attributes of developing LifeEvent as an integrated e-government service. Figure 4.3 is the graphical illustration of the proposed SIE process. It is very important to note that due to the nature of this process, integrated web services (LifeEvents) are built and configured at run-time.

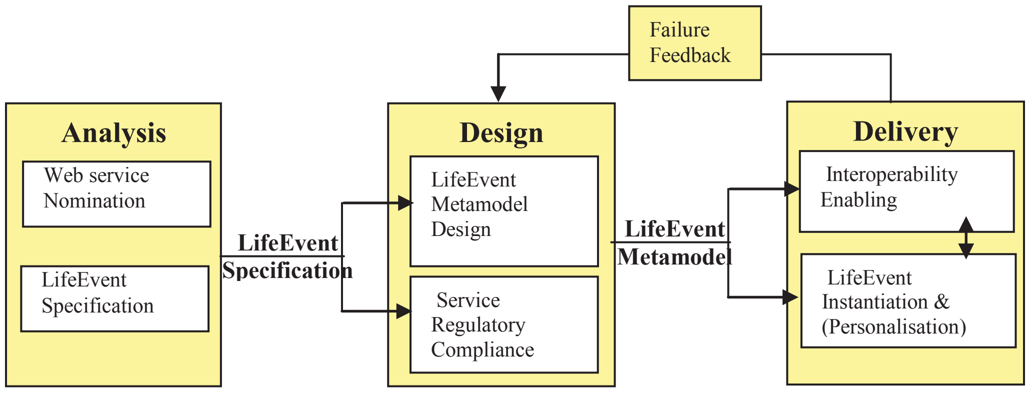


Figure 4.3: Conceptual diagram of the SIE process

4.3.1 Analysis Phase

Designing predictive applications that can make independent decisions or provide effective information for humans to make such decisions requires the use of specific design and modelling techniques. It is widely recommended by the research that such applications make use of the knowledgebase in the form of ontology. It is the position of this research that classical SE is not well-equipped to handle the design of semantically rich and dynamically configured applications.

There are two clear tasks in the Analysis phase in our proposed process, each responsible for producing semantic information at a relevant level of abstraction in the life cycle of a LifeEvent.

The first task is to nominate web services for LifeEvent participation (*Web Services Nomination*). In this task, readily available web services are discovered and registered as potential building blocks for an integrated composite service.

The second task is to propose an integrated service (*LifeEvent Specification*) and indicate its boundaries (scope). The purpose of this task is to specify all the required web services for a specific LifeEvent (all services must have already been deployed and made available by government agencies over the Internet).

First Order Logic is free of all ambiguities associated with natural language expressions. It provides an unambiguous notation for predicate calculus which is one of the best design tools for formalising ontology and deriving axioms for predictive behaviour. The SIE process model described in Figure 4.3 allows for the use of formal ontology description and design tools in this phase.

The result of this phase is two artefacts:

One is an ontological catalogue providing precise semantic information on all the technical communication and QoS parameters of services provided by their WSDL. The role of this ontology (OWL-S) is essential for the LifeEvent in the delivery phase, because it provides descriptors for every atomic service participating in the LifeEvent.

Second is the specification of all the services in the form of LifeEvent Services. This specification is in the form of an ontology data file designed to provide all the knowledge required to understand the highest level of requirements for the proposed integrated web services.

A detailed description of the ontology required in this SIE process model is given in Chapter 5.

4.3.2 Design Phase

Workflow modelling and design can further be divided into generation and specification stages (Liu et al., 2007). The workflow models are considered to be *manual, semi-automatic* or *automatic* depending on the level of semantic and dynamic knowledge representation of the model. The specification of the model, depending on the level of semantic intelligence representation, can be implemented using industry standards such as Business Process Engineering for Web Services (BPEL4WS) or OWL-S. Industry standards such as BPEL4WS are more suitable for static workflows with no semantic intelligence and are entirely configured at design time; however standards such as OWL (McGuinness, 2004) handle specific semantic information that could be used to design a dynamically configurable workflow *(automatic and semi-automatic models)*.

A unified and repeatable e-government service integration process must make use of best practice modelling techniques in a way that increases its reusability in different scenarios, even if these scenarios are in different implementation domains. Our goal in modelling and design is to visualise the LifeEvent design in various levels of abstraction at various phases of our proposed SIE process.

The first task in design phase is therefore to prepare knowledgebase artefacts that can provide knowledge about a more granular level of an abstract LifeEvent. This task (*LifeEvent Metamodel Design*) is to produce an ontology data file that carries all the knowledge required by the system to understand the various details of a LifeEvent Metamodel in order to deliver a specific instance of that LifeEvent.

The second task in the Design phase is the *Service Regulatory Compliance* design, which ensures that every regulatory requirement of every web service that participates in a LifeEvent is understood and enforced as a rule by the LifeEvent designer. If we have chosen a number of web services to participate in a LifeEvent, we must ensure that they are arranged in such a way that there are no prerequisite loopholes or conflicts between the services. For example, if there are three possible web services involved in our *Motorcycle Licence Application* LifeEvent, we must ensure that they are arranged in such a way that the prerequisites of all the services are satisfiable when the LifeEvent is executed.

E-government service integration projects must pay specific attention to *Regulatory* analysis, also possibly to other interoperability issues such as organisational, procedural and jurisdictions. This information is gathered and used as part of the requirements analysis for integrating services. To accomplish semantic interoperability, it is necessary to prepare the service domain ontology

One of the most important areas of analysis and design discussed in this study relates to government regulation and rules. These regulations are to be organised in the LifeEvent ontology to illustrate the semantic correlation of every element. One way to organise the regulatory knowledge is to imbed them in an ontology data file and attach it to the LifeEvent Metamodel when it is prepared and delivered as an item of results from the tasks of this Phase. The other way to approach is to imbed the regulatory knowledge in the LifeEvent ontology as an object property of a concept. In this research we chose the latter approach.

4.3.3 Delivery Phase

The design specifications of automatic workflow models differ from those of static models in that for automatic workflow model. The designer produces a Metamodel that only describes the type of services and the order of execution. This method called Metamodelling, a Metamodel must also contain semantic regulatory information that dictates the terms and conditions of the execution to determine whether it is the right time to execute a particular service and how the results of its execution would affect the overall status of the workflow. A Metamodel produced in the design phase can be used by citizens to create a personalised instance of the LifeEvent to suit their requirements.

The Delivery Phase mainly deals with alternative workflow instances that are produced based on semantic information extracted from LifeEvent ontology and information provided by the customer. The complexity of the situation can go even further by assuming that one of the services may fail at execution time. In this case, the workflow composer must discover the best available alternative service to replace the failed service; this requires automated feedback to the service user and the composition of an alternative workflow instance to substitute for the failed workflow. Our strategy is to facilitate the seamless evaluation of composition candidate services; to improve composability for run-time workflow construction. This requires mechanisms that address the requirements related to ontology, profiles, and their underlying formalism.

Chapter 6 will step through the process of implementing an actual LifeEvent using the SIE phases mentioned here. This will give us the opportunity to evaluate the practical implications of the SIE modelling process.

4.4 Summary

In this chapter we put forward our findings to indicate that the creation of composite service architecture must focus on and represent business objectives, which often concerns producing an added value of delivering a better and more reliable service for users of those composite services.

The complexity of integrating e-government services, especially those developed to an advanced transactional stage requires a highly documented and repeatable methodological approach to ensure the most efficient and reliable transformation of e-government services towards an integrated LifeEvent driven system. In this chapter, an integration methodology to accommodate the integration specific tasks into a typical software engineering process was proposed.

From an engineering process point of view, focusing on the detailed implementation of an integrated e-government delivery system requires a specific attention to tasks and activities that are crucial to a successful service integration project. Hence, this chapter proposed a modified version of the classical software engineering process. SIE has all the tasks and activities that are necessary to handle the complexity of service integration process.

From an implementation point of view, focus on the detailed implementation of our proposed integrated e-government delivery system is also required. Chapter 6 will provide in-depth, detailed information about our proposed framework (ESIM) and Chapter 7 will discuss the implementation platform by developing an integrated e-government service delivery system that can provide a proof of concept for all our theoretical assumptions.

Chapter 5:
LifeEvent Ontology

Before discussing the details of our proposed framework for e-government service integration in Chapter 6, it seems to be helpful and intuitive to describe our design for 'LifeEvent Ontology'. The integration framework makes extensive reference to LifeEvent ontology and it will be easier to understand if we introduce the concept beforehand.

This chapter intends to provide detailed design and a formal description for an ontology that is of crucial importance for our framework, proposed in Chapter 6, to work. It discusses the definition of 'Ontology', 'Service Ontology' and 'LifeEvent Ontology'. Section 5.1 describes the concept of Ontology, then in Sections 5.2 and 5.3 we give details of what the concepts of Service and Service Ontology mean in this thesis. Section 5.4 explains and illustrates the overview of our conceptual design for the LifeEvent ontology. Section 5.5 explains the annotation used in our formal method and we use first order logic to establish the main axioms and design rules for the 'LifeEvent Ontology'.

5.1 Definition of Ontology

In the information technology domain, the term Ontology is a word borrowed from philosophy that refers to the science of describing the kind of entities in the word and how they are related (Smith et al., 2003). According to the Merriam Webster online dictionary, it is defined as[1]:

1) A branch of metaphysics concerned with nature and the relations of beings
2) A particular theory about the nature of being or the kind of things that have existence

In the context of information technology, the most typical kind of ontology has a taxonomy and a set of inference rules (Berners-Lee et al., 2001). A more formal definition of ontology is given by Maedche (2002) as follows:

An ontology is a 5-tuple as $O ::= (C, R, H^C, rel, A^O)$, where C is a set of classes (concepts); R is a set of relations; $H^C \subseteq C \times C$ is called taxonomy, $H^C(c_1, c_2)$ means c_1 "is-a" c_2 ; $rel : R \rightarrow C \times C$ is a function defined for other relations; and A^o is a logical language.

Aside from these definitions, ontologies are used to describe a variety of models, ranging from simplest taxonomies such as Online Directory Project[2] to very complex knowledge models written in first order logic[3].

5.2 RDF and RDF Schema

The Resource Description Framework (RDF) is a recommendation of W3C specifications, originally designed as a metadata model. It is used as a general method for conceptual description or modelling of information that is implemented in web resources, using a variety of syntax formats.

[1] http://www.merriam-webster.com/dictionary/ontology
[2] http://www.dmoz.org/
[3] http://www.ehealthserver.com/ontology/

RDF has an XML-based syntax; it provides a metadata model for ontology knowledgebase and provides a common framework so that applications can process and exchange the information automatically through the World Wide Web.

RDF Schema is the description of RDF language. It is also expressed in XML and provides a simple ontology of RDF concepts and property definitions. RDF provides the basis on which next generation ontology languages are developed. In other words, most new ontology languages are logical extensions of RDF.

5.2.1 RDF Model

RDF identifies objects using URI (Berners-Lee et al., 1998). A URI is a compact series of characters that identifies an abstract or physical resource. It is used extensively as a way to identify resources on the web such as web pages. RDF describes resources in terms of simple properties and property values. Thus, statements in RDF are represented as triples of *(subject, predicate, object).* The subject denotes the resource; the predicate denotes the relationship between the subject and the object; and the object can itself be a resource or a string literal which represents a basic data-type such as Integer, String, or Boolean values.

RDF Schema is a semantic extension of RDF which provides a way to describe how resources in RDF are related to each other. The schema divides resources into groups or *classes* and provides a simple hierarchical classification structure which relates these classes to each another through properties.

The predicate or property element is identified to represent the predicate and object of the statement. Its content is the object of the statement, which is a plain literal. These triples can also be represented as a graph where the nodes are resources (subject and object) and the arcs are the properties (predicates).

5.3 OWL

OWL ontology is also an RDF graph and is represented by a set of RDF triples. As with any RDF graph, an OWL ontology graph has different syntactic forms. OWL extends RDF and RDF schema ontologies by adding more vocabulary for describing properties and classes such as relations between classes, cardinality, equality, richer typing of properties, characteristics of properties, and enumerated classes.

The ontology is not limited to defining the pure hierarchy of classes and their relationships; they are also used to inference class relationships such as equivalence or being disjoint.

5.3.1 OWL Model

Every OWL ontology has a root class called *owlThing*. Every other class defined with the ontology is a subclass of *owlThing*. OWL supports set operators on classes such as union, intersection and complement.

It also allows class enumeration and disjoint. There are two types of simple properties in owl description: one is '*datatype*' and the other is '*object*' properties. Datatype properties are relations between instances of classes and RDF literals or XML schema data types. Object properties are relations between instances of two classes. Properties can also have logical connectivity such as being transitive, symmetric, inverse and functional. As in RDFS, an OWL class contains individuals, which are instances of the class and other subclasses.

Instances are RDF descriptions of a class in which properties are populated with certain values. OWL allows a class to be defined with logical restrictions on certain properties. Classes can be restricted by existential or universal quantifiers. For example, the class *LifeEvent* may have subclasses defined with existential quantifiers on the *hasService* property such as:

hasService some *LifeEventService.*

Thus, a restricted subclass of the *LifeEvent* class can be defined as the *MotorcycleLicenseLifeEvent* class, which contains all LifeEvents that have *LifeEventService* as a service. This amounts to all the instances that have the *hasService* property assigned to the *LifeEventService*. In addition, these properties can also have cardinality restrictions. For example, a *LifeEvent* must have at least one *LifeEventService* but may have more than one. Thus, the *hasService* property can be restricted to:

hasService > 1 and a *LifeEvent* instance can have multiple *hasService* properties.

5.3.2 Sublanguages

OWL has three sublanguages: OWL Full, OWL DL and OWL Lite, all described in (McGuinness, 2004). OWL Lite is the least expressive of them. OWL Lite is somewhat more expressive than RDFS, because it provides simple constraints of classes and properties in addition to supporting a classification hierarchy. OWL DL is modelled on description logics, all conclusions are guaranteed to be computable that means, it supports maximum expressiveness while retaining computational completeness, and all computations will finish in finite time.

OWL DL includes all OWL language constructs. OWL Full is the most expressive of the three sublanguages. The main difference between OWL DL and OWL Full is that in OWL DL, a class is only expressed as a collection of individuals and cannot be regarded as an object in and of itself. However, in OWL Full, a class can be treated simultaneously as a collection of individuals and as an individual in its own right. Due to this difference, OWL Full cannot be checked for soundness using reasoners. A service class will only represent a collection of individuals and does not need to be an individual in its own right, and we would like to use OWL reasoners such as FaCT++ (Tsarkov and Horrocks, 2006) to check for the soundness of OWL documents.

5.3.3 OWL Querying

There are many ways to query OWL ontology. One way is to use a reasoner to create a class with certain restrictions and classify this class within the ontology to see which classes it relates to. The query processor can then look at the equivalent classes, super classes and subclasses to see the different relationships of the query class. Another way is to use the reasoner to reason over the OWL instances. The query processor converts the query to an instance of an OWL class in which all the property values are the same as that of the query. The reasoner then finds all the classes which have this instance as an inferred instance. An inferred instance is one that has not been explicitly instantiated within a class but is inferred to be part of a class because of its properties. The instance is then matched to a number of classes. Aside from these methods, there are a few OWL query languages which have been developed. The OWL Query Language (OWL-QL) (Koponen and Virtanen, 2004) is a formal language and protocol that queries an OWL ontology by finding class relationships. It also allows querying and answering agents to conduct a query-answering dialogue in ontologies represented by OWL.

5.4 Using Ontologies vs. Databases

A logical model of entities and relationships, the Entity Relationship (ER) model in a domain defines a relational database. The ER model can be perceived as a simple ontology because it does not define relationship types and class expressions and thus cannot be classified using ontology reasoners. In addition, an ER model is used for translating to physical tables and thus knowledge about the relationships is captured mainly in the documentation rather than within the ontology itself. This schema is normally specific to the needs of the given application and cannot be shared easily across different domains.

The reasoning power of ontology is the motivation behind its use for representing services rather than using simple attribute-value representations of data, such as in traditional databases. An example of a query which can be done using an ontology which is difficult to do using an SQL query is: ‘Given a service class, find all logically related matches to my query’. Simple Query Language (SQL) also does not support abstract data types, thus making it difficult to determine whether a certain property value belongs in a number of different classes or types. Ontologies can also be shared, re-used and changed. Ontologies can be distributed across the Internet and grow limitlessly, and they can be discovered and shared using their URI.

When new relationships are established within the ontology schema because of ontology migration or the addition of new classes, determining new relationships within the ontology is simply reduced to running a reasoner on the ontology in order to reclassify the classes. For relational databases, changes to the schema may have a fundamental impact on the existing data.

The main drawback to using ontology is that classification is expensive. As ontologies grow large, and especially when instances of classes are stored in the ontology, reasoning becomes a bottleneck. We tackle this problem by storing instances in separate ontology data-files instead of the ontology schema itself. This speeds up the classification process considerably. Also, a positive side-effect of the distributed architecture of LOOSI platform is that it allows each component to handle different ontologies (OWL-S and LifeEvent).

5.5 Web Services Ontology

In order to understand what real-world services are, we look at the work done in the economic and business sciences as well as literature originating from the ICT area.

Hardly any of the generic concepts concerning real-world services show up in current e-commerce product classifications or standards for Web Services, a term that refers to Internet-based technologies, rather than business activities. Web Services are loosely coupled, reusable software components that semantically encapsulate discrete functionality. Web Services, however necessary and useful they are, cannot really be seen as services in the sense of the business science literature; they are currently rather restricted to Input / Output interface specifications.

Further requirements on any service ontology are that its components should be mappable onto configuration task ontologies. This is feasible, because the component based and configuration like nature is inherent to services. In addition, service ontology should be consistent with ontologies that describe value creation in e-government.

There are many initiatives that have made progress towards defining and organising ontologies for web services, two of which have contributed a great deal to the state-of-the-art:

1) OWL-S defines a set of basic classes and properties for declaring and describing services, i.e. an ontology for describing Web Services that enable users and software agents to automatically discover, invoke, compose and monitor Web resources offering services under specified constraints (W3C, 2004). OWL-S tries to cover the description of services in a wide sense, not focusing on a particular application domain or problem.
2) WSMO aims to create ontology for describing various aspects related to Semantic Web Services, but with a more defined focus: solving the integration problem (Domingue et al., 2004). WSMO also takes into account specific application domains (e-Commerce and e-Work) in order to ensure the applicability of the ontology for these areas.

In comparison of the two standards, WSMO covers most of the description elements introduced in OWL-S and introduces additional elements that increase its applicability in real domains. Aspects such as mediation and compensation are key issues to be solved for the realisation of Semantic Web Services, and to make this technology applicable for e-Commerce and e-Work (Zhang et al., 2004). WSMO should provide a higher level of detail for the definition of aspects such as choreography or grounding. If these elements are appropriately covered, WSMO can become a strict superset of OWL-S that also covers relevant issues not covered by OWL-S. WSMO also intends to have an execution platform, called Web Service Modelling eXecution environment (WSMX), while the intentions of OWL-S in this direction are not yet defined.

A web services transaction involves three parties: the service requesters, the service provider, and a mediation infrastructure facility. The service requester, who may broadly be identified as *user*, seeks a service to complete its task; the service provider, which can be broadly identified as *provider*, provides a service sought by the user. The user may not know of the existence of the provider ahead of time, so it relies on mediator infrastructure facilities that act like a service registry and workflow organiser to find the appropriate provider. The role of the mediator registry is to match the request with the offers of service providers to identify which of them is the best match. In Chapters 6 and 7 we will provide the details of such facilities that can act as a delivery platform for LifeEvent and a framework for using such platform. The remainder of this chapter explains the detailed design for an ontology that can provide a foundation for such a framework.

Consider the utilization of a credit card service. A customer can choose the simplest form of a credit card or a more expensive card which offers extra services as free travel insurance, high withdrawal limits, travel assistance abroad, worldwide card replacement in case of loss, linking the card to a preferred supplier and more. Multiple aspects of this service offering can be facilitated by websites: ordering a card, transactions listing, buying other services and goods with the card and more.

Another example is the online organisation of events, such as conferences, board meetings, executive courses, exhibitions, and so on. Their electronic facilitation requires many capabilities, including a good predefined classification of such events, together with a description of their properties, plus the constraints they impose, such as suitable times and spaces (rooms, halls, room setup).

Essentially, ontology is needed that defines the core contents of the service. In addition, electronic facilities should provide the capability to select relevant supplementary services. Such services include travel insurance and high withdrawal limits in the credit card case, and coffee breaks, video facilities, Internet connection, translation, sound, technical assistance, or catering, in the event organisation case. This again occurs in a predefined and standardised, ontology-based way, such that associated additional relationships and constraints can be automatically catered for. Next, customer needs, perceptions and requirements regarding a service usually contain many ‘soft’ statements, leave many things implicit, and often necessitate a significant interpretation and transformation step into the provider's ontological vocabulary and the components that the service provider can actually deliver.

OWL-S has made progress towards configuration-based service composition. We suggest that an important part of a paradigm for the electronic support of real-world services is a generic component-based description of services and what they contain; in other words, a service ontology, such that the electronic design and production of services can be simplified to a configuration task. This task is what is called service composition. In a collaborative e-government scenario, then, the ideal is to have an intelligent support system that:

- Contains ontological descriptions of the service bundle contents;
- Translates customer needs and preferences into suitable terms from the service provider viewpoint;
- Can deal with all the associated constraints in automatically constructing the requested service in a configuration-oriented way, supporting the composition of services from different service providers into a user-controlled work-flow of compound web services.

The biggest limitation of OWL-S is that it only allows for the composition of services (or operations) described in one WSDL. A new system of web services knowledgebase configuration would be necessary if we wanted to compose web services from different vendors. One major challenge is that the service ontology must be sufficiently generic to be useful across many application domains. We discuss below how such ontology might look and present its logical formal description.

5.6 LifeEvent Ontology

This research defines the LifeEvent ontology as the logical extension of OWL-S to provide extended functionality which allows a systematic integration of e-government web services by means of abstraction in design and implementation. The advantage of such an extension is that it preserves and uses all the capabilities inherited from upper ontologies such as OWL and OWL-S, while adding more specialised ontology concepts to achieve precise results for automating the integration of e-government web services by means of abstraction. Our design for LifeEvent ontology will take the goal of OWL-S one step further to integrate atomic and composite services not only from one service provider but from many, to allow the dynamic construction of a user controlled work-flow of readily available web services.

Our model for LifeEvent ontology requires two types of knowledge analysis in order to achieve a more comprehensive solution for the design of the ontology itself. This ontology needs to provide two essential types of knowledge about an event in a citizens' life, as described in Sections 5.8.1 and 5.8.2.

LifeEvent ontology as a service knowledgebase is required to automate the acquisition of individual web service instances in a LifeEvent workflow. It provides service specific information such as availability, service type, service profile, and required communications parameters to the run-time workflow construction process. A service knowledgebase could use multiple ontology descriptors (OWL-S ServiceModel) to obtain the semantic information required by the workflow for the invocation of atomic services. LifeEvent Ontology embodies the following concepts:

5.6.1 LifeEvent Concept

A meta-model that provides a category of knowledge that is needed to answer the question 'In what possible alternative ways can a LifeEvent be constructed?' The answer to this question starts with the concept of *LifeEvent* that is the root element of the ontology inherited directly from the generic concept of *Thing*. This ontology class is the definition of an abstract construct of all possible services that are nominated to collaborate with each other in order to solve a business problem. This ontology class has the inversed functional object property called *hasService.* This object property is of type class *LifeEventService.* The minimum cardinality of this property is one; this means that a LifeEvent must be comprised of at least one service. One of the most important responsibilities of this class is to enforce the rules of government regulations to make sure a legally acceptable workflow is provided that can be instantiated and executed to fulfil a customer request for a service. Listing 5.1 is the RDF code for the construction of this object property.

```
<owl:ObjectProperty rdf:about="le#hasService">
        <rdf:type rdf:resource="&owl;InverseFunctionalProperty"/>
           <rdfs:domain rdf:resource="le#LifeEvent"/>
           <rdfs:range rdf:resource="le#LifeEventService"/>
           <owl:inverseOf rdf:resource="le#describedByLifeEvent"/>
</owl:ObjectProperty>
```

Listing 5.1: Definition of object property hasService

5.6.2 LifeEventInstance Concept

One possible way of implementing a LifeEvent is defined by this concept, meaning that if we consider the LifeEvent as meta-model that only defines the types and the order of possible web services in the workflow, then a LifeEventInstance would be one of the possible ways to create such workflow. This concept has an object property of type ServiceInstance called hasServiceInstance, with the cardinality of one. This property represents the individual invoke-able instances of web services that make up the workflow of the LifeEvent at run time. Listing 5.2 is the RDF code for the construction of this object property.

```
<owl:ObjectProperty rdf:about="le#hasServiceInstance">
   <rdfs:domain rdf:resource="le#LifeEventInstance"/>
   <rdfs:range rdf:resource="le#ServiceInstance"/>
<owl:inverseOf rdf:resource="le#partOfLifeEventInstance"/> </owl:ObjectProperty>
```

Listing 5.2 Definition of object property hasServiceInstance

5.6.3 LifeEventService Concept

This concept provides knowledge about the acceptable web service types and possible sets of actual service instances for every service type. In other words, this concept is the abstract construction of all service types that could potentially be instantiated as a ServiceInstance at runtime. As has been strongly acknowledged by other research literature (Lytras, 2006), the diversity of structures, regulations and procedures affecting networks of heterogeneous administrative units represents a challenge for semantic integration. This type of knowledge is specifically related to e-government service integration, since every LifeEventService participant in any LifeEvent may enforce or be affected by one or more government regulations. These regulations are the governing rules of composite services in the e-government domain, specifically because regulations are one of the integral parts of interagency processes (i.e. where the *LifeEvent* process flow crosses multiple agencies). Furthermore, regulatory knowledge is required for designing an inter-agency workflow that crosses the boundaries of local, state, and federal agencies. It has three object properties:

1) *hasPrerequisite*, that provides the knowledge about the order of services in the workflow or possible required action or

documentation prior to the invocation of the web service. Listing 5.3 is the RDF code for the construction of this object property.

```
<owl:ObjectProperty rdf:about="le#hasPrerequisite">
        <rdfs:domain rdf:resource="le#LifeEventService"/>
                <rdfs:range>
                    <owl:Restriction>
                                <owl:onProperty rdf:resource="le#hasPrerequisite"/>
                                <owl:someValuesFrom rdf:resource="le#Prerequisite"/>
                </owl:Restriction>
        </rdfs:range>
</owl:ObjectProperty>
```

Listing 5.3: Definition of object property hasPrerequisite

2) *hasServiceType*, and
3) *hasServiceSubType* provide knowledge about the type of LifeEventService. Listing 5.4 is the RDF code for the construction of these two object properties.

```
 <owl:ObjectProperty rdf:about="le#hasServiceSubType">
   <rdfs:domain rdf:resource="le#LifeEventService"/>
   <rdfs:range rdf:resource="le#ServiceSubType"/>
   <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
</owl:ObjectProperty>

   <owl:ObjectProperty rdf:about="le#hasServiceType">
   <rdfs:domain rdf:resource="le#LifeEventService"/>
   <rdfs:range rdf:resource="le#ServiceType"/>
   <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
</owl:ObjectProperty>
```

Listing 5.4: Definition of hasServiceSubType and hasServiceType

5.6.4 ServiceInstance Concept

This is a personalised instance of the LifeEventService and provides knowledge about the user preferences at runtime. It implements all the prerequisites of the web service that are enforced by the LifeEventService.

ServiceInstance extends LifeEventService and represents one of many possible runtime instances of the LifeEventService. This ontology class has a property *partOfLifeEventInstance* that points to a class LifeEventInstance described in Listing 5.3 as the inverse functional property of *hasServiceInstance*. It is through this property that we can obtain knowledge about actual web services participating in a LifeEventInstance.

It is our view that any problem a LifeEvent seeks to address can be fitted in to two main areas of concern:

For LifeEvent users, it should describe how to ask for an OWL-S service and what happens when the workflow is being executed. By 'what happens' we mean what are the particular technical and legal requirements of invoking any of the web services in the LifeEvent work-flow.

For managing the workflow, LifeEvent uses a logical description to perform four different functions:

1) To create a composite service work-flow from multiple services in order to perform a specific complex task. It is important to note that '*composing OWL-S services*' as a LifeEvent is different from '*composite services*' described in OWL-S specifications. The *composite services* described in OWL-S are provided by one service provider and specified in one WSDL grounding specification, whereas a LifeEvent composes services from totally different providers with separate WSDL grounding specifications.
2) To manage the status and results of executing a complete or a partial LifeEvent work-flow of web services. This means that a citizen can request a invocation of a LifeEvent but does not necessarily complete the whole LifeEvent in one go. The user also

has the ability to choose to invoke any of the operations listed in a web service descriptor.

3) To coordinate the activities of different service participants (requester, provider and mediator) during the course of the web service enactment.
4) To monitor the execution of the web service and compensate for any web service failures at run-time. This is made possible by the fact that a LifeEvent is a Metamodel and it can be instantiated in many possible ways depending on the availability of different actual ServiceInstances for a particular LifeEventService. There are two object properties of ServiceType and ServiceSubType that can provide essential knowledge about a particular ServiceInstance that enables a LifeEventInstance make a decision on substituting a failed ServiceInstance with a new one that is the closest match in terms of its object properties (*hasServiceType* and *hasServiceSubType*).

The ontology illustrated in Figure 5.1 displays the main concepts of LifeEvent. The abstract concept of LifeEvent is extended by LifeEventInstance. LifeEvent is a 'Metamodel' representing a generic event in a typical citizens' life. This concept points to one LifeEventService. A LifeEventService is the conceptual representation of a web service that holds information about the service type by pointing to an instance of a class ServiceType and an instance of class ServiceSubType. LifeEventInstance is also aware of the position of its corresponding web service within the execution workflow at runtime. LifeEventService represents one element in a possible set of '*n'* elements that makes up the LifeEvent workflow.

A LifeEventService can be instantiated in many ways since it would be pointing to one or possibly more than one LifeEventInstance through ServiceType and ServiceSubType concepts.

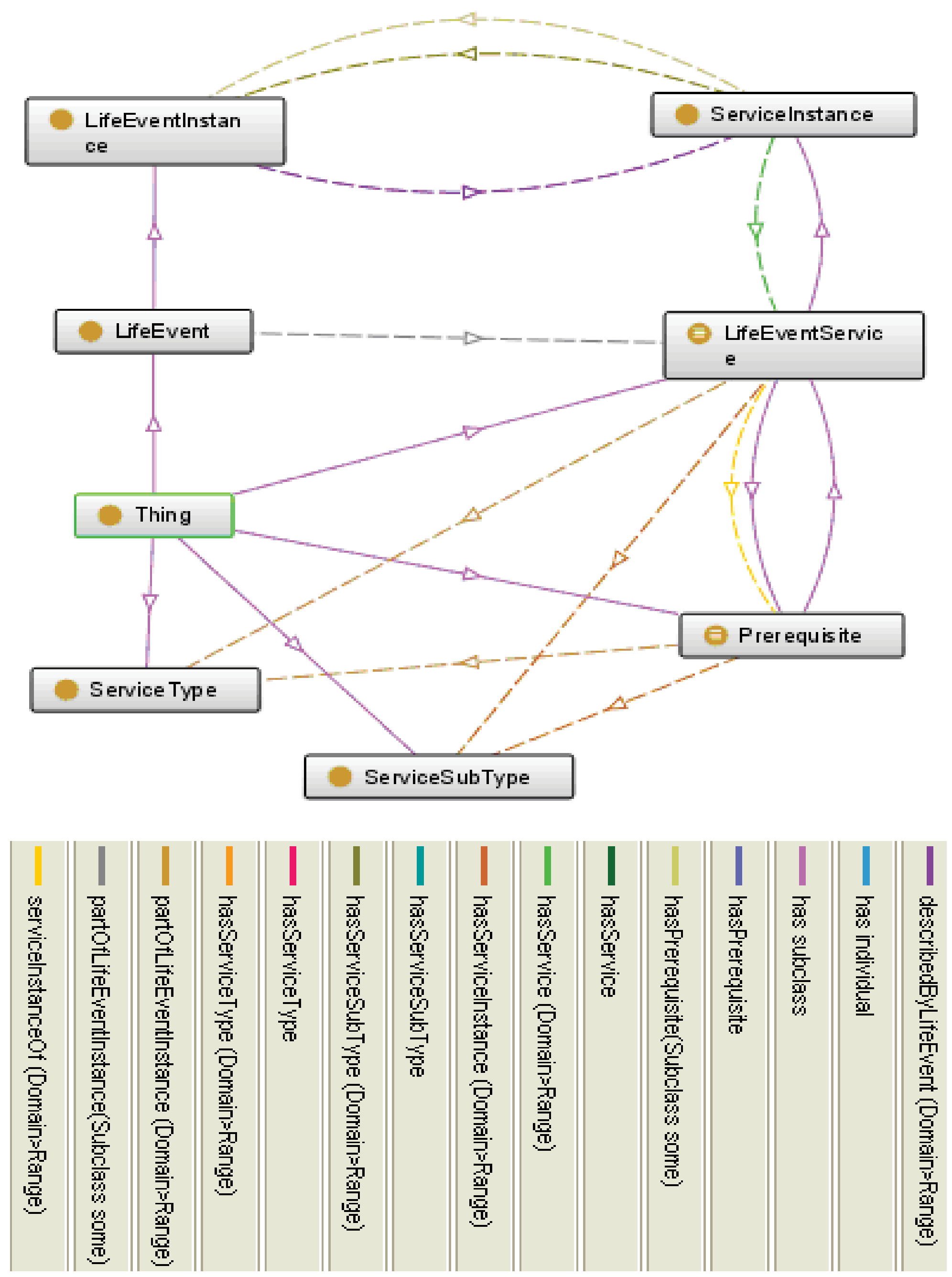

LifeEventInstance
ServiceInstance
LifeEvent
LifeEventService
Thing
Prerequisite
ServiceType
ServiceSubType
serviceInstanceOf (Domain>Range)
partOfLifeEventInstance(Subclass some)
partOfLifeEventInstance (Domain>Range)
hasServiceType (Domain>Range)
hasServiceType
hasServiceSubType (Domain>Range)
hasServiceSubType
hasServiceInstance (Domain>Range)
hasService (Domain>Range)
hasService
hasPrerequisite(Subclass some)
hasPrerequisite
has subclass
has individual
describedByLifeEvent (Domain>Range)

Figure 5.1: LifeEvent ontology conceptual graphs.

In order to lay down legally acceptable foundation for invocation of government web services and to attain a legal outcome of such invocation, one needs to satisfy a set of legally binding regulatory requirements. We achieve this by setting up the rule: 'every element in this workflow must point to at least one prerequisite'. This rule is modelled as concept called 'Prerequisite'. This new concept extends the concept 'Thing', therefore it could represent anything including but not limited to document, payment or, in most cases, another LifeEventService. This rule creates a linked list of services in which every LifeEventService has an object property *hasPrerequisite* which is of type 'Prerequisite'.

It is important to stress that the concept of Prerequisite is very different than the property of 'Precondition' described in OWL-S specifications; this difference is illustrated in Listing 5.5. OWL-S defines the property *Precondition* to be represented as logical formulas like expressions as literals, either string literals or XML literals. The latter case is used for languages whose standard encoding is in XML, such as SWRL or RDF.

```
<Description rdf:about="#process2">
    <hasPrecondition>
        <expr:KIF-Expression>
            <expr:expressionBody>
                (!agnt:know_val_is
                    (!ecom:credit_card_num ?cc)
                    ?num)
            </expr:expressionBody>
        </expr:KIF-Expression>
    </hasPrecondition>
</Description>
```

Listing 5.5: Implementation of property *hasPrecondition* in OWL-S

If an OWL-S process has a precondition, then the process cannot be performed successfully unless the precondition is true. The difference between the *Precondition* properties of OWL-S and the *Prerequisite* concept of LifeEvent falls into two major areas:

1) Consider a process that charges a credit card. The charge goes through if the *Precondition (*card is not overdrawn) is true. If it is overdrawn, the only output is a failure notification. This means that the *Precondition* is an expression of whether to allow the invocation of a service to go ahead, whereas the concept *Prerequisite* of LifeEventService extends the concept *Thing*, therefore it could be anything, including another LifeEventService. A LifeEventService can be invoked if and only if the prerequisite is not itself and the property *isFulfilled* of the prerequisite service is set to true.
2) The other important difference between the two concepts is that the concept of Precondition is confined within the domain of one web service, whereas the concept of Prerequisite goes beyond the domain of one web service and includes a much larger area of the universe of discourse under the domain of LifeEvent. In later subsections we will explain the significant role of the concept *Prerequisite* where it is of the type LifeEventService. Listing 5.6 is a snippet of the owl tag for the class Prerequisite.

```
<owl:Class rdf:about=le#Prerequisite">
        <rdfs:comment>
                This class could constitute anything and is the root element of all
                classes that enforce workflow regulations for the LifeEvent Ontology
        </rdfs:comment>
</owl:Class>
```

Listing 5.6: Class Prerequisite in LifeEvent Ontology

Catering for the possibility of Service substitution in runtime is made possible by the concept LifeEventService, which points to one or more

ServiceInstance; this allows for the substitution of similar web services with some degree of similarity, depending on web service availability or user preferences at runtime.

5.7 LifeEvent Ontology Formal Description

We find it convenient to be able to speak about ontologies as objects and to have a theory of these objects. We will use a first-order language that contains the usual logical operators and symbols: for negation, $\wedge$ for conjunction, $\vee$ for disjunction, $\rightarrow$ for material implication, $\leftrightarrow$for logical equivalence, $=$ for equality ($\neq$ will abbreviate its negation), $\forall$ for the universal and $\exists$ for the existential quantifier. In due course, we will introduce non-logical symbols for the relevant predicates and relations if required. We shall use *x, y, z* as variables ranging over existing entities, and *a, b, c* will be constants denoting such entities. We will use ω, ω' as variables ranging over ontologies, and α, β, γ will be constants denoting ontologies.

We do not offer a full logic, and in particular, there will be no consideration of a deductive system. For the rest, unbound variables are assumed to be within the scope of universal quantifiers.

LifeEvent candidate is an abstract definition of a complex workflow comprised of a number of simple or composite web services. LifeEvent candidate uses OWL-S descriptors of web services nominated by government agencies in conjunction with government regulatory information, which is required to ensure a legal outcome whenever it is executed.

In the following, we provide a formal description for the main concepts (classes) of our LifeEvent ontology followed by their description logic:

LifeEvent = LE
LifeEventInstance = LEI
LifeEventService = LES
ServiceInstance = SI
Prerequisite=PR
ServiceType=ST
ServiceSubType=SST

We shall use the predicate ‘concept’ in order to denote Concepts, thus ‘concept(x)’ is to be read ‘x is a concept’, and a *Concept* can be from any of the following types: (*LE, LEI, LES, SI, PR, ST and SST*). We will use *x, y, z* as variables ranging over concepts.

Symbolically, the first axiom is an existential one, which asserts that there is at least one entity of a certain type. Here we indicate that there exists ontology $\omega$and there exist entity *x* that is of concept (class) LifeEvent in a variable ontology ω.

$$\exists\omega,\exists x\left[\Omega(\omega)\wedge LE(x)\right]$$

We use the predicate Ω in order to denote token, thus ‘$\Omega(\omega)$’ is to be read ‘ω is an ontology’. An instance of a given ontology token α is an entity whose existence is recognised by α. We will write ‘inst(*x*, α)’ which is to be read ‘*x* is an instance of α’. Hence, there is no empty LifeEvent ontology.

$$inst(x,\omega)\rightarrow\left[consept(x)\wedge\Omega(\omega)\right]$$

In addition any existence is a constituent of ontology.

$$\Omega(\omega)\rightarrow\exists x\left[consept(x)\wedge inst(x,\omega)\right]$$

The predicate ‘realises’ denotes the instances of associated *LES* concepts within the LifeEvent ontology. While each instance may be a model on its own, a combination of LESs may be aggregated to constitute a composite model. In that case, the services are considered to be the components of a model. *LES* (*y*) play role (*r*) that require skills (*s*) needed to perform their role.

$$\exists y[LES(y) \wedge (plays(y,r) \wedge has(r,s))]$$

Given that *LES* provides a set of specific operations O and the *x* is a variable over this set to fulfil a *t* that is a variable over the set of tasks T, it would be required to have a subset S’ from the skill set of S.

$$[LES(y) \rightarrow (provids(y,x) \wedge (O \vee S'))]$$

We must ensure that every *LES* carries enough semantic information to facilitate the runtime reconfiguration in case of a web service failure during the *LEI* execution. Every *LES* has an object property ‘*hasPrerequisite*’ that makes one *LES* the prerequisite service of the value of this property in the workflow of *LESs*. Each of these object properties point to another *LES,* essentially creating a linked list of *LESs* in which every *LES* is aware of its place in the list_through the data property called *workflowPosition*. In other words, a LifeEvent is the construct of a two dimensional linked list, in which the first dimension is the list of meta-services and the second dimension is the list of service instances for each meta-service.

5.8 Summary

In this chapter, we outlined our design to extend the OWL-S in order to propose LifeEvent ontology as the executable unit for composing e-government web services. We introduced an ontology that accommodates the concept of LifeEvent within the process of e-government web services composition. The idea used in this ontology design introduces an innovative approach towards the whole process of e-government web service integration and delivery by shifting the focus of e-government integration towards citizen-centric view of the e-government development.

We put forward a formal design for an ontology knowledgebase to manage the workflow of integrated web services in a linear manner. However, we also recognise that more research is required to specify and formalise the design of more complex types of web services composition such as parallel service processes in complex workflows.

In Chapter 6, we will propose a framework (Sanati and Lu, 2008, Sanati and Lu, 2009b, Sanati and Lu, 2009a) known as ESIM which is a repeatable modelling framework that formulates efficient and repeatable sets of tasks performed in a configurable order to achieve the objectives of the e-government service integration project. The design of this framework is based on the theoretical constructs of Service Integration Engineering methods introduced in Chapter 4 and the extensive use of LifeEvent ontology introduced in this chapter.

Chapter 6: E-government Service Integration Modelling Framework

It is widely perceived that one of the most fundamental responsibilities of any methodology is to support repeatability. Many research projects are currently attempting to formulate new modelling strategies for web services composition as the new technologies and development techniques are proposed in the SOC (Huhns and Singh, 2005) domain. In the absence of documented best practices of design and modelling techniques for web services composition, it would be greatly beneficial to discover such common practices and formulate a repeatable methodology for such projects. This research recognises e-government service integration as a sub-domain of web services composition and this chapter describes the detailed specifications of a repeatable modelling process for e-government service integration projects. E-government Service Integration Modelling (ESIM) framework is a repeatable process, defining required tasks and activities to provide effective management control over the e-government service integration process. Chapter 6 uses the ESIM framework to model a simplified example of an e-government integration solution as a proof of concept.

6.1 Semantic Information Integration

The research community and IT industry have experimented with many solutions, ideas and modelling theories for integrating web services, some of which advocate the use of semantic/ontology and some of which specialise in integrating data and transaction aspects of web services. It is the position of this research that e-government services cannot be integrated in one ‘Big Bang’, therefore an automated dynamic process is required to enable the gradual integration of e-government services in an intelligent way. This could reduce the risk of failure associated with the traditional resistance of governments to technology adoption and budgetary issues such as increasing costs of data and process integration.

Semantic aspects of service integration in the e-government domain are an active research subject in the international scientific community. In addition to the examples described in Chapter 2, several other research papers address this issue and are worth specific mention. For example Graciela Brusa (2007) discusses how information integration must be performed through the use of e-government-specific ontologies. Zhang (2008) proposes the use of grid technology for the storage and retrieval of diverse information in the context of e-government. Zhou (2005) proposes semantic grid-based information integration. In this work, fundamental problem of information integration is addressed by using the de-centralisation properties of grid technology. Barnickel (2006) propose the use of cross-ontology semantic web service composition to achieve interoperability in e-government.

Unlike most aforementioned solutions, this research advocates a client-centred approach in e-government service integration. Therefore, the basic design strategy is to create a system that *enables the ordinary citizens to* *create their own personalised composite services and enact them in a controlled environment*. Chapter 7 is dedicated to discussing the detailed design and implementation of a software system to validate the conceptual assumptions of ESIM framework.

6.2 Concept of Abstraction

The concept of abstraction borrowed from an object-oriented paradigm is used to represent complex data structures. Abstract objects or data structures can form hierarchical representations to provide easy-to-understand solutions for complex models. Abstraction is the means by which only a certain level of detail is exposed by the entity, depending on the level of representation intended for specific models. This study invokes the principle of data abstraction in the context of e-government services to represent LifeEvent as a composite service in different levels of granularity (abstraction), from very little detailed abstraction to higher levels of detail about its underlying service structure and business rules in different levels of granularity. The ESIM framework discussed in this chapter describes such a conceptual model.

6.3 ESIM Overview

A Citizen-Centric model of e-government integration requires government organizations to restrict their delivery of services, to be more closely aligned to the needs of constituents rather than internal business processes. E-government is a critical component to this Citizen-Centric view of government as these strategies allow service users to have more control over how and when they receive services. More than simply deploying web-based solutions, e-government strategies must aim to digitalize interactions between governments and citizens, businesses and other government organisations in order to provide more timely, personalised and comprehensive integrated service.

Similar to private businesses, improving user satisfaction is also a top goal for federal, state, and local e-government strategies.

The ESIM framework is a repeatable modelling framework that formulates an efficient and repeatable sets of tasks performed in a sequential order to achieve the objectives of e-government service integration. This framework is constructed based on the assumptions of Hybrid E-government Integration Model to achieve a Citizen-Centric process for intelligent delivery of e-government integrated services. Figure 6.1 illustrates an overview of the process flow in ESIM framework, which is explained in detail later in this chapter. The design of this framework is based on the theoretical constructs of Service Integration Engineering methods introduced in Chapter 4 and the extensive use of LifeEvent Ontology introduced in Chapter 5.

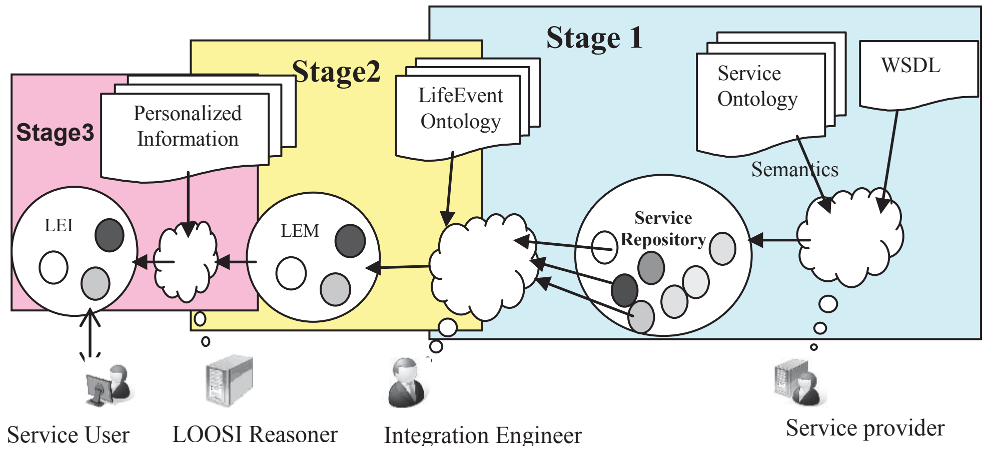


Figure 6.1: Overview of the ESIM framework.

The automated nature of ESIM framework will reduce the aforementioned risks by allowing gradual and incremental participation of government agencies for integration of their web services. Based on an incremental use of the ESIM framework government agencies can decide which services to integrate and when.

6.4 ESIM Framework

The process flow of ESIM framework is meant to be of a Control-oriented type using deadlock resolution, exclusion, concurrency, and process activation and deactivation. Flowcharting, being one of the primary process-oriented modelling techniques is therefore the most suitable modelling technique to used to model the requirements of this type (Thayer and Dorfman, 1977).

Figure 6.2 illustrates the LifeEvent life cycle from the technical perspective, begins with the initiation of a LifeEvent candidate at Stage 1 through to the proposed Metamodel at Stag 2 and finally the execution of a personalized LifeEvent instance at Stage 3, each in deferent levels of abstraction.

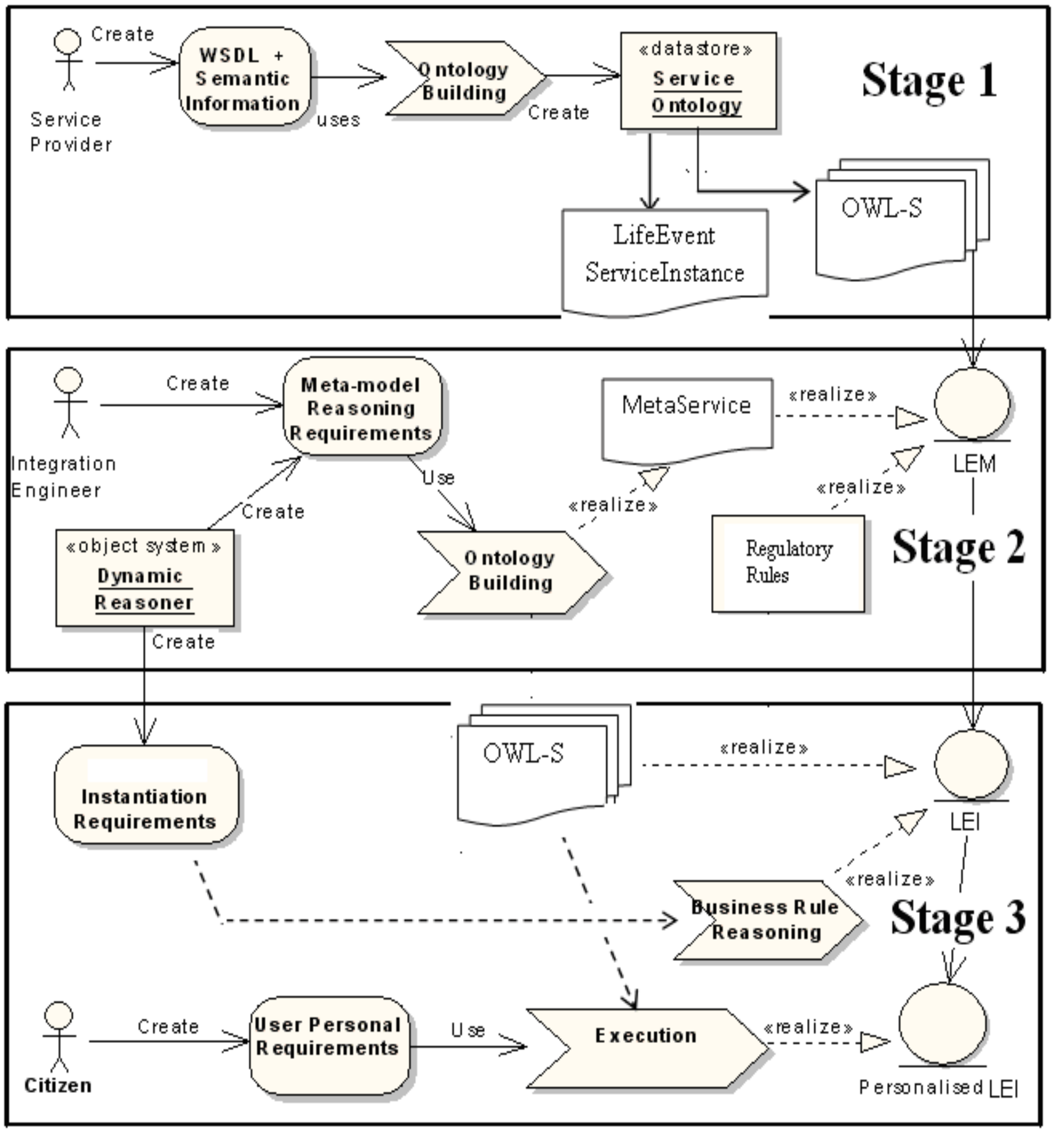


Figure 6.2: ESIM framework for LifeEvent Life Cycle

This model is a component modelling view of the ESIM framework with different types of stakeholders and their requirements. The framework is split into three main stages along the logical implementation, to help better understand its main activities. The ESIM framework requires

LifeEvent to go through three stages in its life cycle before reaching service consumers. These three stages are summarised as follows:

6.4.1 Stage 1 - Translating WSDL to OWL-S

For an e-government service integration system to work needs to create a repository of available web services. Stage 1 is designed specifically to populate such a repository with what is in fact a collection of *Service Ontology Instances,* generated from already published and available web services. In this stage nominated web services are registered in the service repository and an instance of OWL-S is created for every registered web service.

Inputs at this stage consist of:

1) The specifications of the participating services, such as a web services deployment description in the form of WSDL.
2) Complementary semantic information about the services such as service profile information, quality of service and service category. This information is needed to form a semantic information structure about a specific service individual in the OWL-S schema.

Outputs of this stage are:

1) An instance of OWL-S Service. The diagram in Figure 6.3 illustrates an example of how the service provider agent can perform a fusion of the static data provided by WSDL with semantic information provided by service owners or service publishers into an instance of OWL-S. This function is called service registration, during which the service provider nominates a web service for participation in LifeEvents.

2) A LifeEvent ServiceInstance. In this example ESIM combines the syntactic information from WSDL with semantic information to produce LifeEvent ServiceInstance data files as well as OWL-S ontology data files.

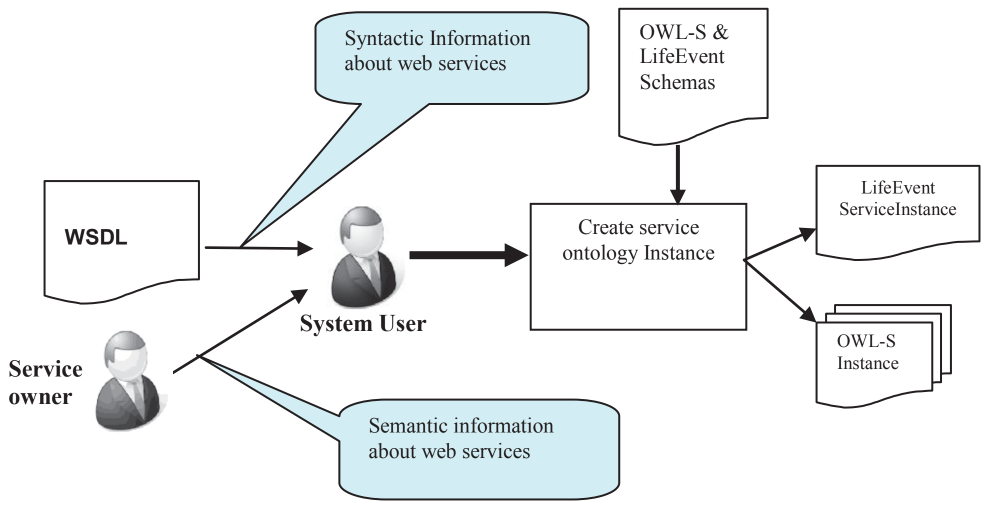

Syntactic Information about web services
OWL-S & LifeEvent Schemas
WSDL
System User
Create service ontology Instance
LifeEvent ServiceInstance
OWL-S Instance
Service owner
Semantic information about web services

Figure 6.3: LifeEvent ServiceInstance and OWL-S construction process.

6.4.2 Stage 2 – LifeEvent Meta Modelling

The review of recent literature in Chapter 2 implicitly indicates that the cost of data and process integration has been a well known obstacle of integrating web services. We propose an automated dynamic process to enable a gradual and incremental integration of e-government services in an intelligent way. This is important, because the system will not know, for example, what services may be available prior to the instantiation and execution of a LifeEvent (runtime). The automated nature of ESIM framework is intended to reduce the cost of data and process integration by allowing the gradual and incremental integration of web services. This is made possible by introducing the concept of the LifeEvent Metamodel (LEM).

Designing predicative applications that can make independent decisions or provide effective information for humans to make such decisions requires the use of specific design and modelling techniques. Predicate calculus is one of the best design tools for formalising ontology and deriving intelligent algorithms. OWL has proven to be a very powerful tool of this type, enabling semantic reasoning in web applications. OWL is currently the main technology for implementing semantic web applications and therefore, the capabilities of semantic software engineering (Sheu and Kitazawa, 2007) are required to handle semantic reasoning design problems.

The design specification of automatic workflow models differs from those of static models in that the designer of an automatic workflow model has to produce a model that only describes the service type and the order of execution. This model is in fact a metamodel used to generate and configure a number of possible variations in run-time. The ESIM framework makes use of all service, regulatory and domain semantic information related to the Metamodel and its running instance for the run time configuration of LifeEvent instance. The ontology information dictates the terms and conditions of the execution in order to determine, whether it is the right time to execute a particular service and how the results of such an execution would affect the overall status of the workflow and the user profile. Figure 6.4 is the conceptual illustration of a Metamodel and its run time instantiation.

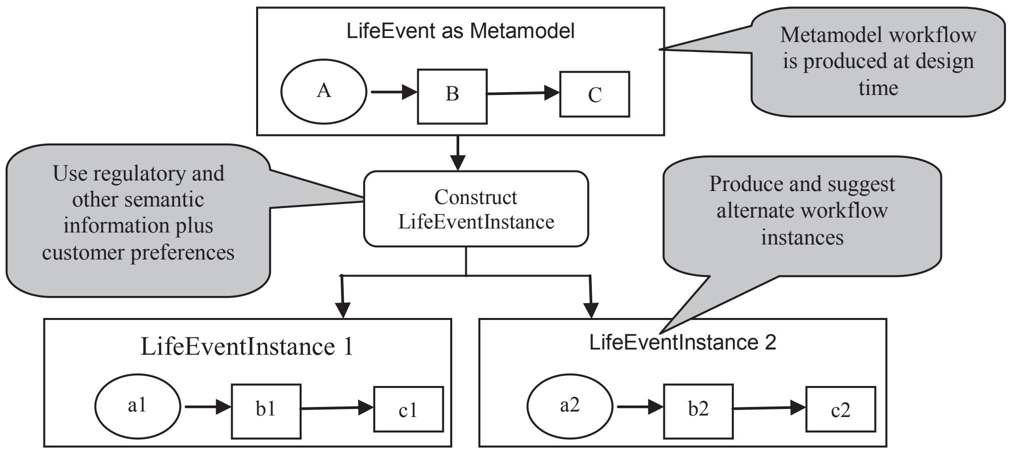

LifeEvent as Metamodel
A
B
C
Metamodel workflow is produced at design time
Use regulatory and other semantic information plus customer preferences
Construct LifeEventInstance
Produce and suggest alternate workflow instances
LifeEventInstance 1
a1
b1
c1
LifeEventInstance 2
a2
b2
c2

Figure 6.4: Metamodelling conceptual diagrams

Based on our design specification, a LEM is a workflow construct of an individual LifeEventService (LES) that indicates the type and subtype of web services to be used in a composition. Every MetaService (MS) holds a set of possible SIs that can alternatively be used in an execution of the workflow at runtime. Assuming a LEM has *n* MS. which is shown in the following definitions:

Definition 1.

Let $LEM = \{M_i | i = 1,2,....n\}$, and $M_i = \{I_{i_j} | j = 1,2,....j_k\}$,

where *LEM* is a workflow construct of individual,
MS that indicate the type of web services to be used and M_i is the MS *number* '*i*' in the *LEM,* and I_{i_j} is the SI number '*j*' of the *MS* numbers '*i*'.

Definition 2.

Let each MS (M_i) consist of (i_k) *SI* so that for any $I_{i_j} \leftarrow M_i, \{i = 1,2,\ldots,n \wedge j = 1,2,\ldots,k\}$, let *x* and *y* be 2 instances of *j* in a LEM, we call I_{i_x} is prerequisite of I_{i_y} and note $I_{i_x} \succ I_{i_y} \leftrightarrow I_{i_y} = I_{i_{x+1}}$.

We also assume each SI is a prerequisite of itself to achieve recursive search functionality when looping through the SI set of (M_i) to determine the order of execution. The latter concept is expressed in the description 3 as follows:

Definition 3.

$I_{i_x} = I_{i_y} \leftrightarrow (I_{i_x} \succ I_{i_y} \wedge I_{i_y} \succ I_{i_x})$

Definitions 1, 2 and 3 have paved the way for defining our recursive function of instantiating and executing *MS* in the LEM workflow as follows:

Definition 4.

Let R_{I_i} be an ordered set of *SI*, then $R_{I_i} = \{R_{i_j} \mid R_{i_j} \succ R_{I_i}\}$.

In the Stage 2 of ESIM framework, a LEM is created. Inputs of this stage are:

1) Regulatory specification that provides the governing rules for the workflow of LEM.
2) A set of Meta Services that will be used to construct a LEM.

The output of this stage is an instance of LifeEvent ontology that is a logical extension of OWL-S, semantically enriched with e-government services regulatory information. The instance is represented in the form of a Metamodel specification called LEM. In this activity, an integration agent can construct a LEM. This activity heavily relies on regulatory rules and the Meta Services specification. Figure 6.5 illustrates the activity in which an integration engineer will create or edit a LEM instance.

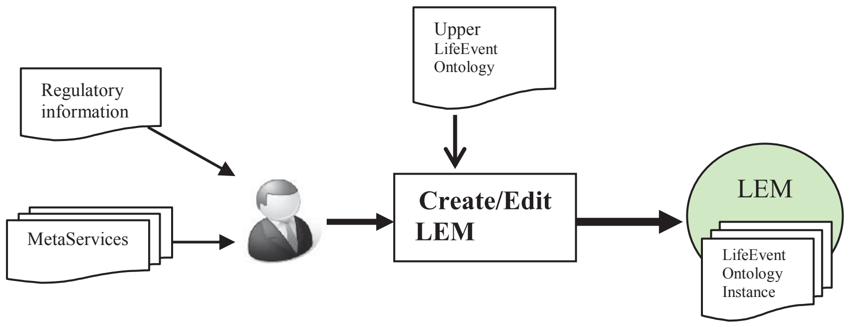


Figure 6.5: Creating/Editing LEM

6.4.3 Stage 3: LifeEvent Instantiation and Execution

This is the LifeEvent Instantiation and execution stage; in this stage the third level abstraction of LifeEvent is created. An executable LifeEventInstance (LEI) is created based on its LEM upon request by a service consumer. The requirements of this stage consist of user preferences data including domain information and service personalisation. The input of this stage is a LEM specification from Stage 2 and the LifeEvent ontology instance associated with the LEM that is required for run time reasoning. The output of this stage is a personalised executable instance of LEI.

The final stage of the LifeEvent life cycle in ESIM framework is illustrated in Figure 6.6, in which the e-government service users issue an execution request for LEI.

In this stage the system analyses the user's request specifications in conjunction with the LEM workflow requirements to infer the correct execution decisions. In this stage:

- Appropriate available service instances are selected,
- Regulatory rules are applied and,
- A service user profile is constructed in order to instantiate and executes a personalised composite service workflow or what is referred to as 'Personalised LEI' in this thesis.

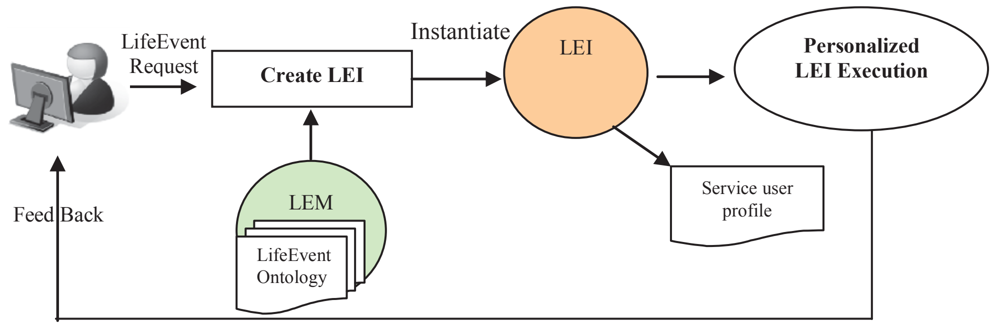


Figure 6.6: Runtime workflow construction and LEI execution

A LEM retains semantic information that dictates the terms and conditions of its execution. It determines whether it is the right time to execute a particular service and demonstrates how the results of its execution would affect the overall state of the workflow. In Figure 6.4 we show a LEM, which consists of three MSs namely (A, B and C), where each MS has a set of SIs. Our design strategy facilitates the automated evaluation of services that are nominated for composition, taking advantage of other models developed in the community (Soon, 2002). We understand that improved composability for runtime workflow construction requires mechanisms that address the ontology requirements, profiles, and their underlying formalism (Lu et al., 2006), therefore composite service Metamodel for a LifeEvent is designed to fulfil those requirements and the following are specifications of their generic model.

A LEM workflow only indicates the type of an atomic service nominated for composition, as well as the order of execution. The specification of the instance of individual services is configured dynamically at runtime. For example, where a workflow is designed to use a service of type ‘driving school’, there may be *x* numbers of different driving school services available at any given time. A rule engine can use available ontology information to support a decision in which a specific service such as a riding school service located in the suburb of Wyong in NSW Australia can be used at runtime given the customer requirement parameters and current state of execution.

A LifeEvent must be instantiated by a service user (customer or citizen) from an already existing LEM based on user specific preferences. When the LifeEvent instance is executed at runtime, there is always a possibility (for whatever reason) that the user would like to replace a service with a similar match with different attributes, or in the case of a failed service system, may need to reconstruct the LEI by replacing one or more services with the closest similar match. In order to measure the degree of similarity between the two instances of the same concept, we define the ‘instance distance’ to measure instance-level similarity.

We suppose two instances (*i* and *j)* are the same concept, where A and B represent their sets of the data property respectively.

Definition 5. The Instance distance between *i* and *j,* denoted by *SI(i,j)* is defined as:

$$SI(i,j)=\begin{cases}0 & A\equiv B,\\ ID(A,B) & A\subseteq B\vee B\subseteq A,\\ ID(A,B)+P & A\not\subset B, B\not\subset A,\\ \infty & A\bigcap B=\perp\end{cases}$$

Where $(\mathrm{i}=1...\mathrm{n}\,,\,\mathrm{j}=1...\mathrm{n})$ and *ID(A,B)* is the quantified distance between the two instances and *P* is a penalty item always given a positive number (i.e. 2). For example if the properties of ServiceInstance SI_1 is defined as $SI_1(a,b,c,d,e,f)$ and the properties of SI_2 is defined as $SI_2(d,e,f,g,h)$,

Therefore instance distance between the two can be denoted as $ID(SI_2\subseteq SI_1)$.

We use TF-IDF based methods for measuring data property similarity of the two instances. TF-IDF based methods use a vector space model (Baeza-Yates and Ribeiro, 1999), treating strings as 'bag of tokens' and ignoring the sequential order of tokens in the strings. A data property that consists of one or more strings can be viewed as a virtual document containing a bag of string tokens.

If there are *N* properties, the corresponding *n* virtual documents form a virtual quantity, which may finally have a vocabulary of *n* distinct tokens. A sparse *n*-dimensional token vector *Vi* *(i=1...n)* can be derived from the *i*-th virtual document with each element $v_{i,j}$ having TF-IDF value computed as follows:

Definition 6. An element of a virtual document is described as:

$$v_{i,j} = TF_{i,j} \times \log \frac{N}{DF_j} \quad \text{Where } (\mathrm{i} = 1 \ldots \mathrm{n}\,,\ \mathrm{j} = 1 \ldots \mathrm{n})$$

Where $TF_{i,j}$ is the frequency of token t_j in the *i*-th document and DF_j is the frequency of the document that contains the token t_j. For two instances with *A* and *B* as their derived token vectors respectively, the similarity between them is computed as a normalised dot product between their corresponding token vectors (Wang et al., 2008).

Definition 7. The similarity between *S* and *T* is measured as:

$$SIM(S,T) = \frac{\sum_{j=1}^{n} s_i . t_j}{\|S\| . \|T\|} \quad .$$

Given the LifeEvent ontology schema, we can construct a hierarchy of the concepts by computing the subsumption relations between the concepts using an ontology reasoner such as FaCT++; this allows us to explore the 'instance-level similarity' of concepts. Because ontology data is contributed by different information sources (i.e. government agencies) separately, the quality and the focus of completeness of the data may vary. However, instances should have some relationships according to their common ontology schema.

For example, if two instances from different information sources are identified as instances of concept SI, we can compute the context similarity between instances by reasoning with the properties and checking the similarities of data property values. The computational complexity of equation 6.3 may be high in general. However, this computation is only expensive if both *A* and *B* property sets have a lot of members. However, based on the design of the LifeEvent Ontology (Chapter 5), a LifeEventService may only have handful of properties and not many alternative instances would exist in the vector of the MS. Therefore, the value of the computational complexity will not grow significantly.

6.5 Summary

This chapter proposes a repeatable process for e-government service integration, which is mostly overlooked in the relevant literature. This research recognises the lack of a unified common practice for e-government service integration projects in the e-government domain. The goal is therefore not only to propose enabling tools and technologies but also to introduce an innovative approach towards the whole process of e-government service integration. In this chapter we proposed the concept of LifeEvent abstraction as a unit of requirement for integration and delivery of e-government services. We introduced the integration framework that can support the roll of concept 'LifeEvent' within the process of e-government service integration. The concept of LifeEvent is used to encapsulate the requirements of an integrated complex e-government service composed from multiple web service provided by possibly many web service providers.

The LifeEvent model is also designed for ordinary citizens to create and invoke their own personalised LifeEvent. For this to happen, the LifeEvent had to be a Metamodel rather than a solid model, So that it would be flexible enough to allow users to instantiate it in every possible way.

In the following chapters we will show how the proposed ESIM framework can work in practice.

Chapter 7: LifeEvent Ontology Oriented Service Integration Platform

E-government service delivery software design process has mostly been technology-centred rather than citizen-centred. This has led to a series of problems. With the growing number of e-government services available, citizens have to bear the burden of remembering more service locations, login names and passwords. There is a lack of standard mechanisms to centrally manage these services and provide integrated single access capability to multiple applications. It is not unusual that users experience long response times or failures, especially with transactional operations. A fundamental shift in the design process is thus required to radically improve software usability.

In this chapter we discuss the detail analysis, design and implementation of LOOSI platform prototype, which is a novel software application to provide the functionalities required to support web services integration modelling processes called the ESIM framework. This framework as discussed before is a citizen-centric web service integration modelling framework that can allow e-government service users to design their own personalised composite web services and execute them in a safe and controlled environment.

In Chapters 4, 5 and 6 we introduced SIE, LifeEvent Ontology and ESIM framework as the prerequisite contributions that can enable and support the technology discussed in this chapter. In this chapter we present the detailed specifications and implementation of a new software architecture based on workflow and decision support to fulfil the requirements of the ESIM framework.

In the reminder of this chapter we will explain the LOOSI platform prototype in details from the following different prospective:

1) Technology overview (Section 7.1)
2) Architectural overview including software logical components of the system (Section 7.2).
3) Application detailed design, including software physical components (Section 7.3).
4) Application functional design, which includes software & application functional design and logical pathways of the main functions (Section 7.4).

7.1 Technology Overview

In the technology design for the prototype we use semantic web ontology in order to achieve interoperability in service integration and delivery. This involves the use of domain-based ontology with reasoning capability in order to dynamically create the required new workflows of web services in run time. From methodological point of view the proposed framework (ESIM) enables gradual and incremental integration of readily available web services, by allowing the automated transition of semantically diverse web infrastructure in to an integrated environment. This method will enable all government agencies to participate in implementation of the prototype in a gradual and voluntary basis.

Service integration engineers can use the instructions given by the ESIM framework to achieve a semi-automated integration of disparate web services in to workflows with a minimum risk and effort. The proposed integration framework will therefore dramatically reduce the cost of service integration by allowing agencies to decide not only which services to integrate but also when to integrate.

The prototype is designed and developed to deliver the functionalities promised by the ESIM framework. This software application, acts as a delivery vehicle for e-government integrated services. This platform makes use of LifeEvent ontology as its knowledgebase to perform the decision making on dynamic integration and reconfiguration of LifeEvent services at run-time. It provides a physical facility for implementation of the ESIM framework to achieve a systematic and repeatable process for integrating the e-government web services.

On the technology note, this research has found Service Oriented Architecture (SOA) to be an emerging premier integration and architectural approach in contemporary complex and heterogeneous computing environments. Figure 7.1 is a generic illustration of this architecture. Since there is a great potential for an SOA system to be built on open standards and can be realised using web services, developing meaningful web service has become an important requirement for SOA applications. Therefore, it can be said that using open source software components is an integral part of SOA strategy.

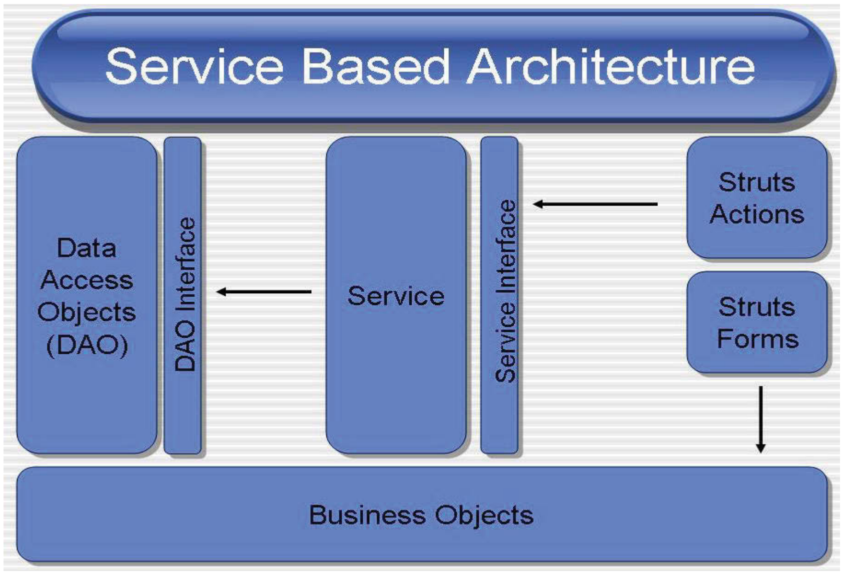


Figure 7.1: Service Oriented Architecture overview (Thomas and McGough, 2009)

7.2 Architecture Overview

In this section we describe the detailed architectural design of the prototype. This architecture adopts a web based enterprise application, specifically SOA development (Siew, 2006).

High level of interoperability and accessibility has a significant importance for a service integration system that is designed to discover, compose and deliver readily available web services to the service consumers. The prototype is designed based on the SOA architecture, and it utilises the maximum versatility of using open source software applications and persistence. It makes use of enhanced reasoning capabilities of the semantic web through the extensive use of ontology in order to achieve more intelligently manageable dynamic integration of web services.

In Chapter 5 we argued that the reasoning capability of ontology is more useful when it's used as knowledgebase for the semantic applications, In the rest of this chapter therefore we describe a simple design solution for this purpose, that can take advantage of reasoning capability of ontology knowledgebase. This design allows for dynamically constructed yet user controlled workflows to be configured and executed at run time by using an ontology knowledgebase. The use of ontology reasoning makes the prototype a dynamically configurable system with a higher degree of personalisation capability. It is this personalisation capability that gives the users the freedom of constructing their own LifeEvent workflows. The diagram in Figure 7.2 is the overall architectural view of the prototype that describes the main components of the system.

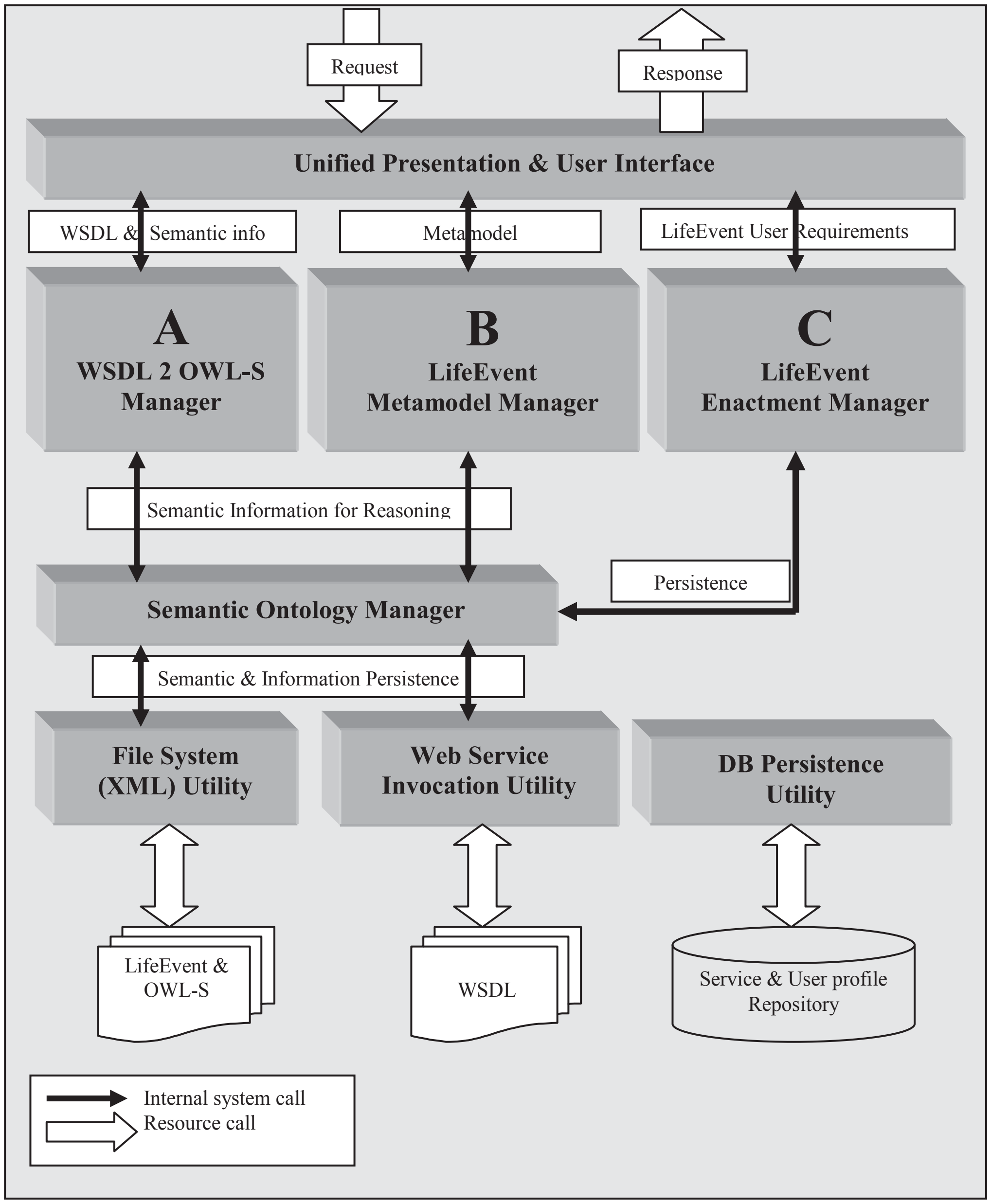

Request
Response
Unified Presentation & User Interface
WSDL & Semantic info
Metamodel
LifeEvent User Requirements
A
WSDL 2 OWL-S Manager
B
LifeEvent Metamodel Manager
C
LifeEvent Enactment Manager
Semantic Information for Reasoning
Persistence
Semantic Ontology Manager
Semantic & Information Persistence
File System (XML) Utility
Web Service Invocation Utility
DB Persistence Utility
LifeEvent & OWL-S
WSDL
Service & User profile Repository
Internal system call
Resource call

Figure 7.2: Architectural overview of LOOSI platform prototype

Functional capabilities of the prototype is categorised and put in to three major components, each designed to provide one-to-one functional support for the corresponding stages in the ESIM framework as follows:

1. ***WSDL2OWL-S*** is a component that provides the functional support needed for creating OWL-S ontology data, based on the process prescribed by the Stage 1 in the ESIM framework that was described in Section 6.4.1.
2. ***LifeEvent Metamodel Manager*** component corresponds to the Stage 2 of the ESIM framework enabling the integration engineers to create LifeEvent Metamodels from available web services. This component uses the OWL-S ontology data files that were created in Stage 1 of the ESIM described in Section 6.4.2.
3. ***LifeEvent Enactment Manager*** is a component designed to enable LifeEvent users to create their own personalised instance of the LifeEvent Metamodel on the fly. It also provides the facility to invoke individual web services within a process workflow. This component corresponds to the Stage 3 of the ESIM framework in Section 6.4.3.

There are many other components in this system that are mostly facilitators of the aforementioned main components, some important ones are described as follows:

Unified Presentation & User Interface is a presentation layer based on Struts technology. It is designed to render the user graphical interface for all the functionalities of the prototype. This component is capable of producing the GUI elements required to capture the necessary data for invocation of web services. This intelligent capability is realised by its ability to translate the input/output parameters information of the web service obtained from the WSDL.

Semantic Ontology Manager is a component comprised of all java classes required by the system to provide ontology related reasoning, interrogation and editing functionalities.

File System Utility provides the persistence functional utility such as read and write for both OWL-S and LifeEvent ontologies.

Web Services Invocation Utility provides end point functional utility for dynamically invoking and communicating with published web services in the internat.

DB Persistent Utility provides the functional connection point for relational database persistence to store edit and use intermediate data required by the system to continue operations.

7.3 Design Pattern Overview

The LOOSI application prototype has adopted web enterprise application architecture based on Model-View-Controller (MVC) design pattern, which is illustrated in Figure 7.3. MVC is a software architecture, currently considered a popular design pattern used in software engineering. The most important advantage of this pattern is that it isolates domain logic (the application logic for the user) from the user interface (input and presentation), allowing independent development, testing and maintenance of each layer (separation of concerns).

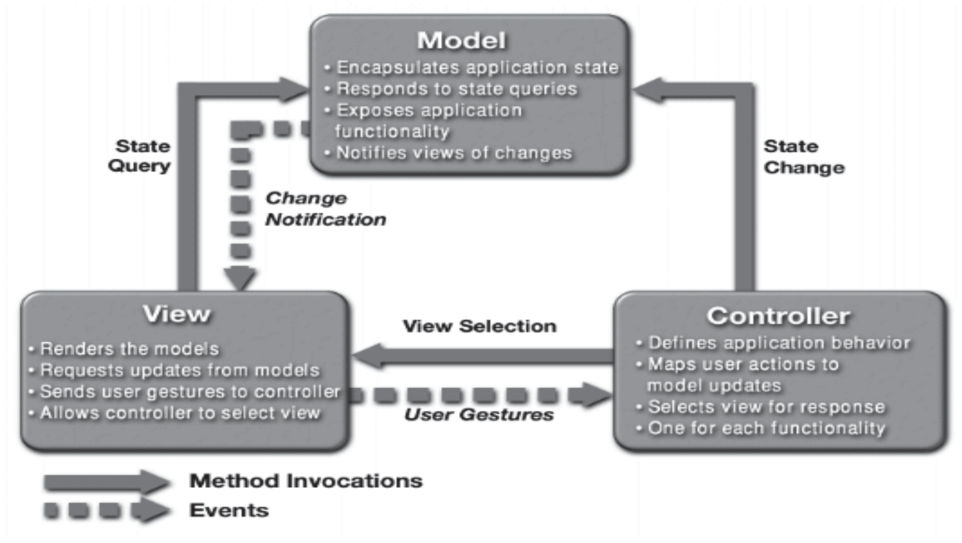

Model
• Encapsulates application state
• Responds to state queries
• Exposes application functionality
• Notifies views of changes
State Query
Change Notification
State Change
View
• Renders the models
• Requests updates from models
• Sends user gestures to controller
• Allows controller to select view
View Selection
User Gestures
Controller
• Defines application behavior
• Maps user actions to model updates
• Selects view for response
• One for each functionality
Method Invocations
Events

Figure 7.3: MVC design Patterns (SunMicrosystems, 2002)

The MVC architecture originally applied to map the traditional input, processing, and output tasks to the graphical user interaction model. MVC is currently used straight forward to map these concepts into the domain of multi-tier enterprise applications.

1) **Model** - The model represents enterprise data and the business rules that govern access to and updates of this data.
2) **View** -The view renders the contents of a model. It accesses enterprise data through the model and specifies how that data should be presented.
3) **Controller** - The controller translates interactions with the view into actions to be performed by the model.

The prototype is configured as per MVC design pattern specifications. This application uses JSP technology in its presentation layer to generate the views of graphical user interface. We use action classes based on

Apache Struts (Apache, 2007) specifications as controller components that trigger JSP view via Struts configuration files. We also put in place a data service layer called Data Access Object (DAO) to play the role of data facilitator for controller classes. DAO layer will handle all data access functionalise and transaction controls, passing around java objects as data models. We use Open Source database MySQL version 5.0 for the persistence layer to store user profile, temporary and intermediate state of incomplete LifeEvents.

Figure 7.4 is a visual illustration of the prototype implementation according to MVC design pattern.

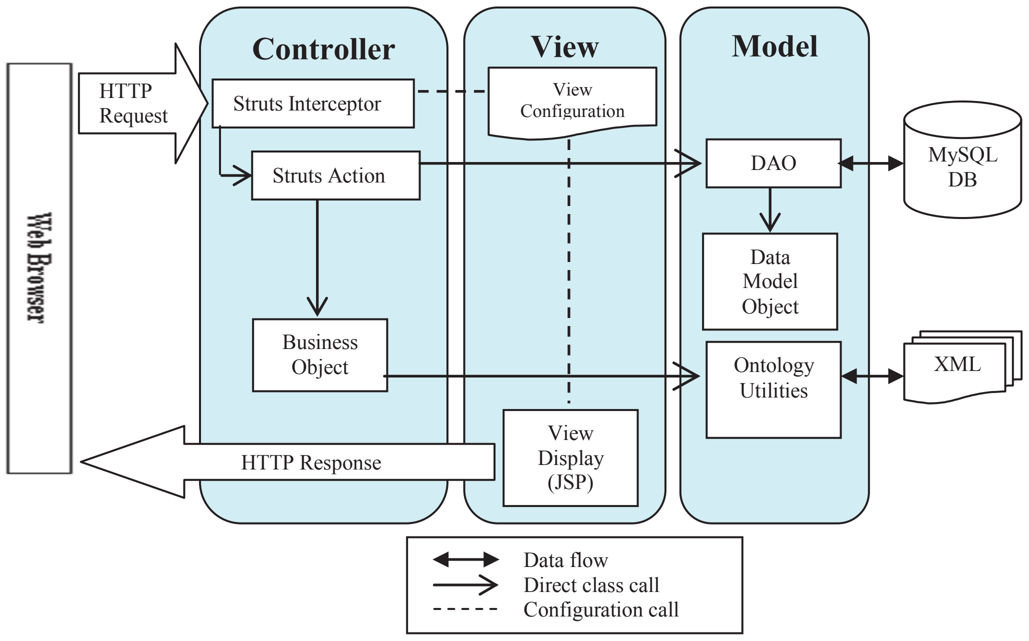


Figure 7.4: The prototype compliance of MVC design Patterns

7.4 Object Oriented Design

SOA and MVC design pattern require the implementation design of the target system to be based on Object Oriented Design (OOD). Therefore in this section we describe the class/object level design based on OOD. Figure 7.5 is a summary class diagram of main classes in the system mapped in to the MVC design pattern; we use java language for the implementation of the system.

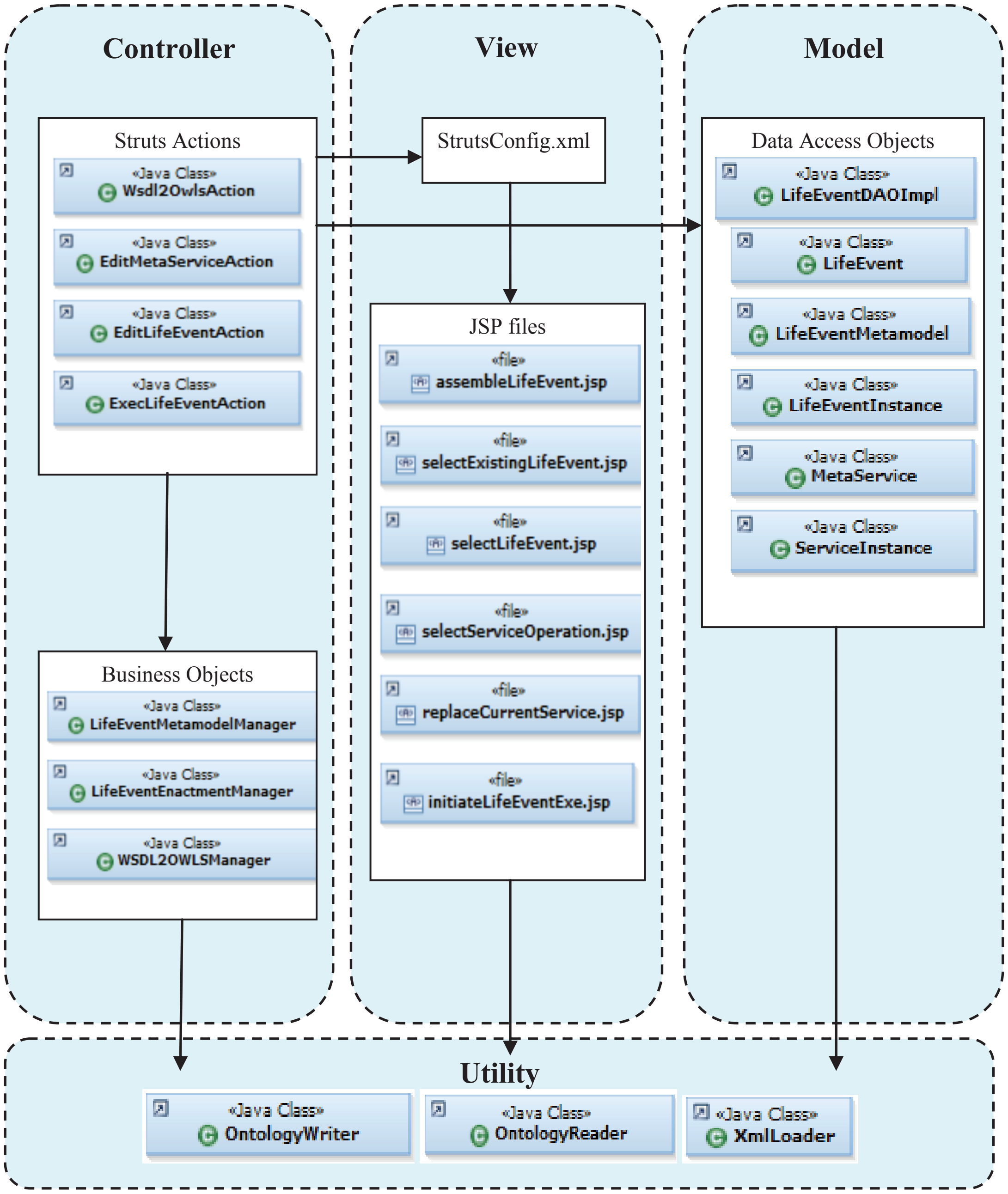

Controller
View
Model
Struts Actions
«Java Class» Wsdl2OwlsAction
«Java Class» EditMetaServiceAction
«Java Class» EditLifeEventAction
«Java Class» ExecLifeEventAction
StrutsConfig.xml
JSP files
«file» assembleLifeEvent.jsp
«file» selectExistingLifeEvent.jsp
«file» selectLifeEvent.jsp
«file» selectServiceOperation.jsp
«file» replaceCurrentService.jsp
«file» initiateLifeEventExe.jsp
Data Access Objects
«Java Class» LifeEventDAOImpl
«Java Class» LifeEvent
«Java Class» LifeEventMetamodel
«Java Class» LifeEventInstance
«Java Class» MetaService
«Java Class» ServiceInstance
Business Objects
«Java Class» LifeEventMetamodelManager
«Java Class» LifeEventEnactmentManager
«Java Class» WSDL2OWLSManager
Utility
«Java Class» OntologyWriter
«Java Class» OntologyReader
«Java Class» XmlLoader

Figure 7.5: The LOOSI prototype java classes mapped in MVC

The Struts action classes in the Controller layer, provide controller functionality for the system. There is one action class for each major function in the system.

The *WSDL2OWLSAction* is the controller class for registering web services and generating the OWL-S ontology. The functional code that facilitates the ontology generation is provided by the class *WSDL2OWLSManager*. The *EditLifeEventAction* and the *EditMetaServiceAction* classes are designed to work with the *LifeEventMetaModelManager,* they collectively enable the user to create a LifeEvent Metamodel, which corresponds to an ontology individual of type LifeEvent. The third Action class is *ExecuteLifeEventAction*, this is a controller class for creating personalised instances of LifeEvent Metamodel and executing the ServiceInstances in a user controlled environment.

At the base of the diagram in Figure 7.5 there are a number of utility classes that provide support for ontology and user information persistence.

7.5 Application Component description

In this section we discuss the design of the prototype in more details and describe how it uses LifeEvent ontology to achieve semantic interoperability in service integration and delivery.

Design of predictive applications that can make independent decisions or provide effective information for users to make such decisions requires the use of specific design & modelling techniques. First Order Logic (FOL) of predictive calculus is the best design tool for formalising ontology and consequently driving intelligent algorithms from them since it is free of all ambiguities associated with natural languages. This research uses two effective tools to express the theorem behind its design, and that is through both graphically and logical expressions.

1) First we use flowchart diagrams to illustrate main logical pathway that describes the functional requirements of the component in question,
2) Second we use FOL to express the implementation of any algorithm in a logical expression language whenever a complex operation is required.

Before we describe the details of our design lets first introduce some notations to simplify the presentation of the proposed algorithms. These notations are summarised in Table 7.1.

Variable Name	Meaning
LE	A set of LifeEventServices that make up the LifeEvent Metamodel
$\overline{M}$	Number of members in LE
LEI	A set of ServiceInstances that make up the LifeEventInstance
$\overline{I}$	Number of members in LEI
P	A set of web service operations that make up OWL-S process
ST	A set of ServiceTypes described by the LifeEvent ontology
PR	A set of Prerequisites of a ServiceInstance

Table 7.1: Notations for web service enactment component functional description

7.5.1 WSDL2OWL-S Manager

WSDL2OWL-S is designed to facilitate the registration of readily available web services in prototype for government agencies and other businesses. These registered services later can be used to construct LifeEvent Metamodels. This intelligent component is used to create a set of OWL-S service ontology data files (profile, service, process and grounding) that can collectively provide necessary knowledge to the

LifeEventMetamodelManager. In turn LifeEventMetamodelManager will be able to create the LifeEvent Metamodel based on its design requirements. This component is designed to provide support for the Stage 1 of the ESIM framework.

WSDL2OWL-S component uses the information in a WSDL of a web service and combines it with other semantic information provided by the web service provider (government agency or a business). The product of this activity is a set of OWL-S service ontology data files (profile, service, process and grounding) for the web service in question. Then the newly created OWL-S ontology data is stored in the repository of the LOOSI prototype and is used throughout the system for various purposes including generating LifeEventInstances and web service enactments. Diagram in Figure 7.6 is the illustration of the main functional pathway in WSDL2OWL-S component.

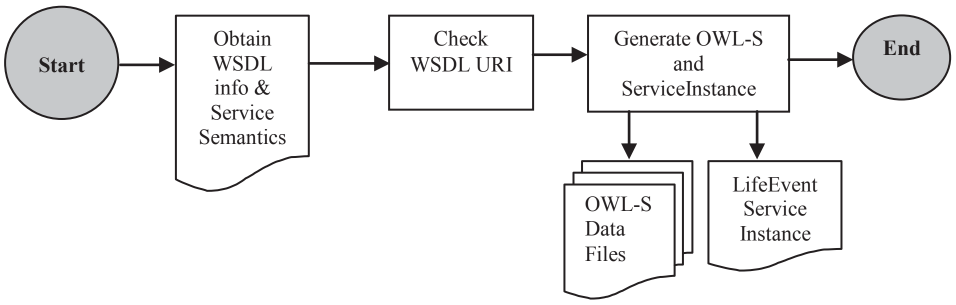

Start
Obtain WSDL info & Service Semantics
Check WSDL URI
Generate OWL-S and ServiceInstance
End
OWL-S Data Files
LifeEvent Service Instance

Figure 7.6: WSDL2OWL-S component functional flowchart

7.5.2 LifeEvent Metamodel Manger

LifeEvent Metamodel Manger is an intelligent component that can create LifeEvent Metamodels based on design requirements of a citizen life event as it is preserved and agreed by the prototype system administrators.

It uses the OWL-S service descriptors stored in a repository, which was described in Section 7.3.1.

The main function of this module is to enable the system user to construct a LifeEvent Metamodel. The system represents the LifeEvent Metamodel data as a Java object, which is an equivalent Java class to the ontology concept of LifeEvent. The strategy is to facilitate an automated evolution of the LifeEvent Metamodel in to an executable workflow of web services in an iterative environment to improve composability for run-time workflow construction. Workflow construction requires mechanisms to address the requirements that pertains ontology, profiles, and their underlying formalism.

The formal design of the main decision point of LifeEventMetamodel Manager is illustrated in Figure 7.7 and described as follows:

Consider the situation where a LifeEvent Metamodel is consists of a set of Meta Services where each MetaService can be a Metamodel of many possible service instances and in turn each service instance may be a model on its own.

A combination of services may be aggregated to constitute a composite model. In that case Meta Services are considered to be the components of a Metamodel. In this Metamodel the position of every Meta Service is determined by its prerequisite.

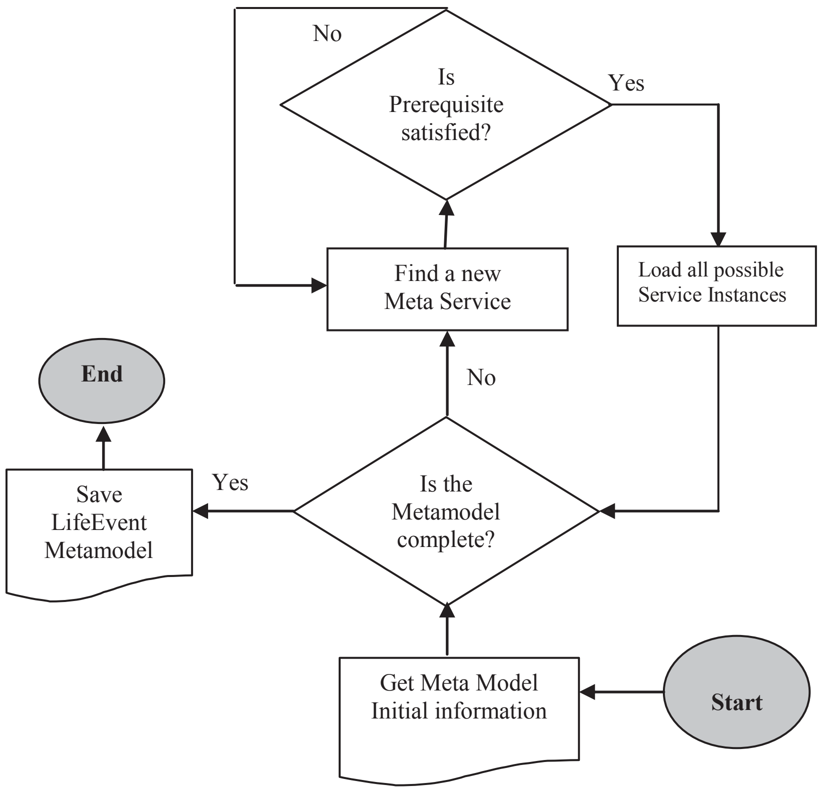

No
Is Prerequisite satisfied?
Yes
Find a new Meta Service
Load all possible Service Instances
End
No
Save LifeEvent Metamodel
Yes
Is the Metamodel complete?
Get Meta Model Initial information
Start

Figure 7.7: LifeEvent Metamodel Manager functional flowchart

7.5.3 LifeEvent Enactment Manger

LifeEvent Enactment Manger is an intelligent component designed to generate LifeEvent instances based on one LifeEvent Metamodel and specific service user requirements.

LifeEvent is modelled at compile time (Stage 3 of the ESIM), which is called LifeEvent Metamodel and specified as follows:

LifeEvent Metamodel will only indicates the type and position of the services and the order of invocation in an integrated workflow, not the instance of individual services (i.e. the workflow uses a service of type 'driving school', there may be *x* number of different school services available at any given time).

LifeEvent Enactment Manager uses available ontology knowledge to decide which specific service to use in runtime, given the customer requirement information. Hence a conceptual alignment between a LifeEvent Metamodel and its new context (runtime implementation instance) LifeEventInstance is established for meaningful integration.

In this section we present the main logical pathway of LifeEvent Enactment Manager, first in the form of Data Flow Diagram (DFD) illustrated in Figures 7.7, and then we describe the formal description of the program logic using FOL description language.

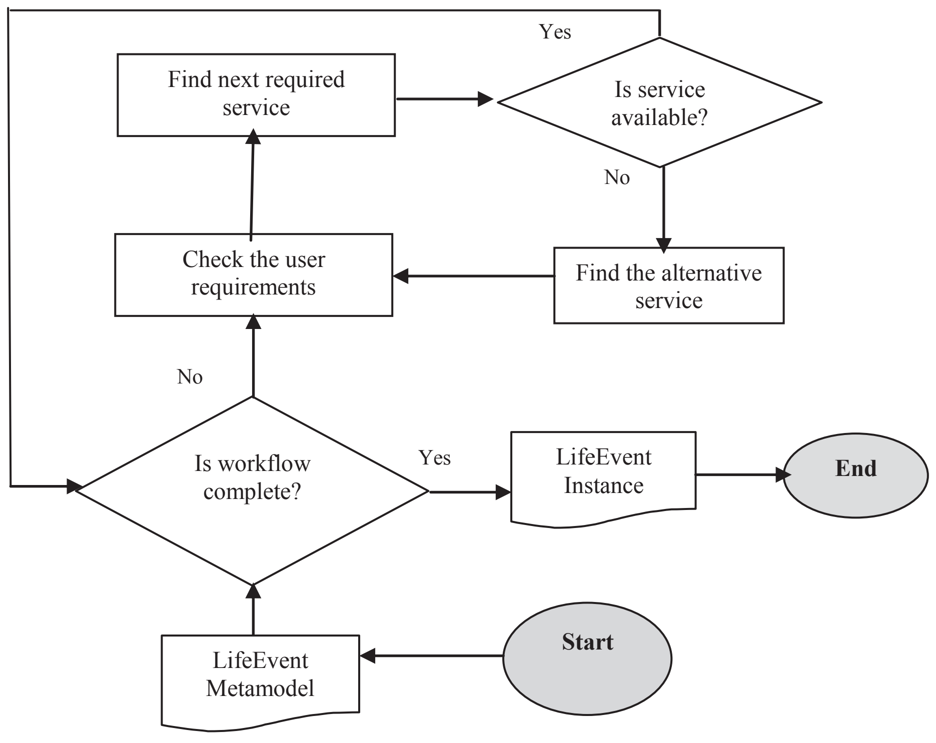


Figure 7.8: LifeEvent Enactment Manager functional flowchart

The two main decision points in LifeEvent Enactment Manager are illustrated in Figure 7.8 and described as follows:

Mapping every member from LE to a possible set of members in LEI based on ST such that the ST for every ServiceInstance is the same as the ST for the LifeEventService. Formal description if the program logic is as follows:

Do While $\exists m\left(LE(m) \wedge \exists i(LEI(i) \wedge i_t \subseteq ST \wedge i_t \equiv m_t\right)$

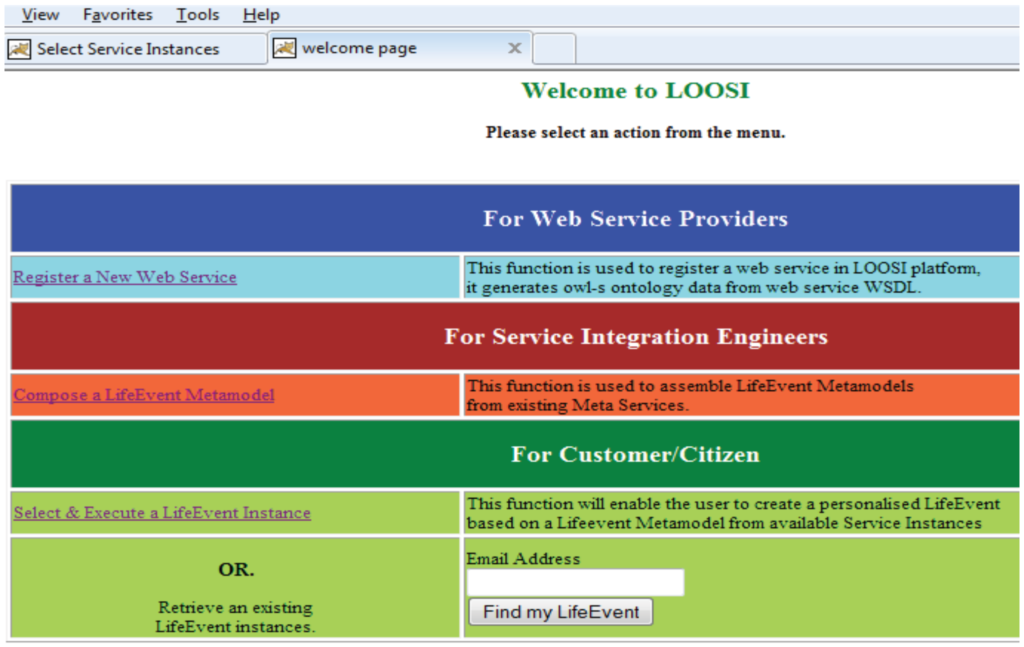
And $\neg\left(\overline{M} = \overline{I}\right)$

7.6 Application Development

This section describes the GUI of the prototype and the development efforts that resulted in creating a software application that is used to test the ESIM framework.

GUI for the System is specifically designed to reflect the main three stages of ESIM framework, therefore providing a seamless translation of the process flow in to the actual functionality of the application. For example the entry page of the application prototype is clearly conveying the ESIM process model. Figure 7.9 is showing the entry page of the web application that displays the three main functionality of the system in a form of a link that is to invoke the function in distinctly coloured cells with a brief description of them next to the link.

Complete walkthrough of the application prototype is described in Chapter 8 as an experimental evaluation of the software.



Welcome to LOOSI

Please select an action from the menu.

For Web Service Providers

Register a New Web Service	This function is used to register a web service in LOOSI platform, it generates owl-s ontology data from web service WSDL.

For Service Integration Engineers

Compose a LifeEvent Metamodel	This function is used to assemble LifeEvent Metamodels from existing Meta Services.

For Customer/Citizen

Select & Execute a LifeEvent Instance	This function will enable the user to create a personalised LifeEvent based on a Lifeevent Metamodel from available Service Instances
OR. Retrieve an existing LifeEvent instances.	Email Address

Figure 7.9: Three main functions of the prototype

One of the most important advantages of LOOSI prototype is that it hides all the complexities of the semantic web application behind the user friendly and intuitive interface. Chapter 8 will show an experimental run on the prototype. This experiment will go through all functionalise of the system displaying the results in screen shots, and then will compare all the actual and expected results.

7.7 Summary

In this chapter we described the design and implementation of a novel software system, used as an integration and delivery platform (LOOSI) for integrating e-government services. This prototype makes use of LifeEvent ontology as the knowledgebase to assist in decision making on dynamic integration and reconfiguration of LifeEvent services at run-time. The prototype is designed based on the SOA architecture, it utilises the maximum power of using open source software applications and persistence. This software application is designed to adopt enterprise application architecture based on MVC design pattern. The prototype allows service providers to register their readily available web services. The results of this registration are a set of service ontology data files and an instance of a LifeEventService. Ontology data created during service registration provides the bases for the creation of a LifeEvent ontology that is a metamodel form of LifeEvent. Finally all the ontology data produced during the service registration and LifeEvent Metamodel creation can be used by citizens to create a personalised instance of LifeEvent and executed to solve a business problem.

Chapter 8: Framework Evaluation and Experimentation

In this chapter, we present an evaluation of the proposed LifeEvent ontology introduced in Chapter 5 and the LOOSI platform prototype introduced in Chapter 7. Evaluation in Chapter 7 will implicitly validate the design assumptions of Chapter 6. We conduct this evaluation in two parts. In Section 8.1 an implementation of LifeEvent ontology presented and the value of its complexity is measured against the OWL-S ontology schema. Through a comparative analysis of OWL-S with LifeEvent ontology schema we arrive on an understanding of the efficiency of our design for LifeEvent ontology. In Section 7.2 we run through an implementation of the prototype presented by displaying a test run of the software and present the results of this execution.**LifeEvent Ontology Evaluation**

There is no restriction on the complexity of the logic that may be used to state the axioms and definitions of concepts in ontology. The distinction between terminological and formal ontologies is one of degree rather than kind. LifeEvent ontology tends to be smaller than terminological ontologies, but its axioms and definitions can support more complex inferences and computations. We conduct the experimental evaluation of the LifeEvent ontology in two stages. We use the ontology editor tool Protégé to design and develop the LifeEvent ontology schema as the preparation for evaluating the LifeEvent ontology. We use the FaCT++ reasoner plug-in from within Protégé to perform structural validation of the schema. To evaluate the efficiency of LifeEvent ontology an experiment is conducted to measure the complexity of the ontology through a set of well known formal methods and demonstrate the results in a numerical as well as graphical representation. We compare the LifeEvent ontology to OWL-S ontology since it's the most conceptually similar to it.

8.2 Ontology Measuring Methods

As ontologies grow in size and number, it is important to be able to measure their complexity quantitatively. Quantitative measurement of complexity can help ontology developers and maintainers better understand the current status of the ontology, therefore allowing them to better evaluate its design and control its development process. We are using a suite of ontology metrics (Zhang et al., 2009), at both the ontology level and the class level, to measure the design complexity of LifeEvent ontology. This ontology complexity measurement metric was evaluated in an empirical analysis on public domain ontologies to show the characteristics and usefulness of the metrics. The proposed metric suite is useful for managing the LifeEvent ontology development projects.

8.2.1 Ontology Level Metrics

We use three different ontology level metrics to measure the complexity of the ontology:

Size Of Vocabulary (SOV) Measures the amount of vocabulary defined in ontology. Given a graph representation $G = (N, P, E)$ of an ontology, where N is a set of nodes representing classes and individuals; P is a set of nodes representing properties; and E is a set of edges representing property instances and other relationships between nodes in the graph G. In this measurement *SOV* is defined as the cardinality of the named entities N_n and P_n in *G*: $SOV = |N_n| + |P_n|$, where N_n representing named classes and individuals, and P_n representing user defined properties.

Edge Node Ratio (ENR) measures the connectivity density. In this measurement *ENR* tends to increases as more edges are added between nodes. The greater the *ENR*, is the greater the complexity of an ontology. *ENR* is calculated as follows:

$ENR = \frac{|E|}{|N|}$, as the division of the number of edges $(|E|)$ by the number of nodes $(|N|)$.

Tree Impurity (TIP) measures how far ontology's inheritance hierarchy deviates from being a tree and it is defined as being:

$TIP = |E'| - |N'| + 1$, where $|E'|$ is the number of *subclass* edges and $|N'|$ is the number of nodes in an ontology's inheritance hierarchy.

8.2.2 Class Level Metrics

These metrics are mostly concerned with the class level specific statistics, the most popular technique in this method is known as ***Number of children (NOC)***. To calculate *NOC* for a given class C, *NOC* measures the number of its immediate children in the ontology inheritance hierarchy given, as follows:

$$NOC_c = \#\{D | D \in N' \wedge (D, rdfs: subClassOf, C) \in E'\},$$

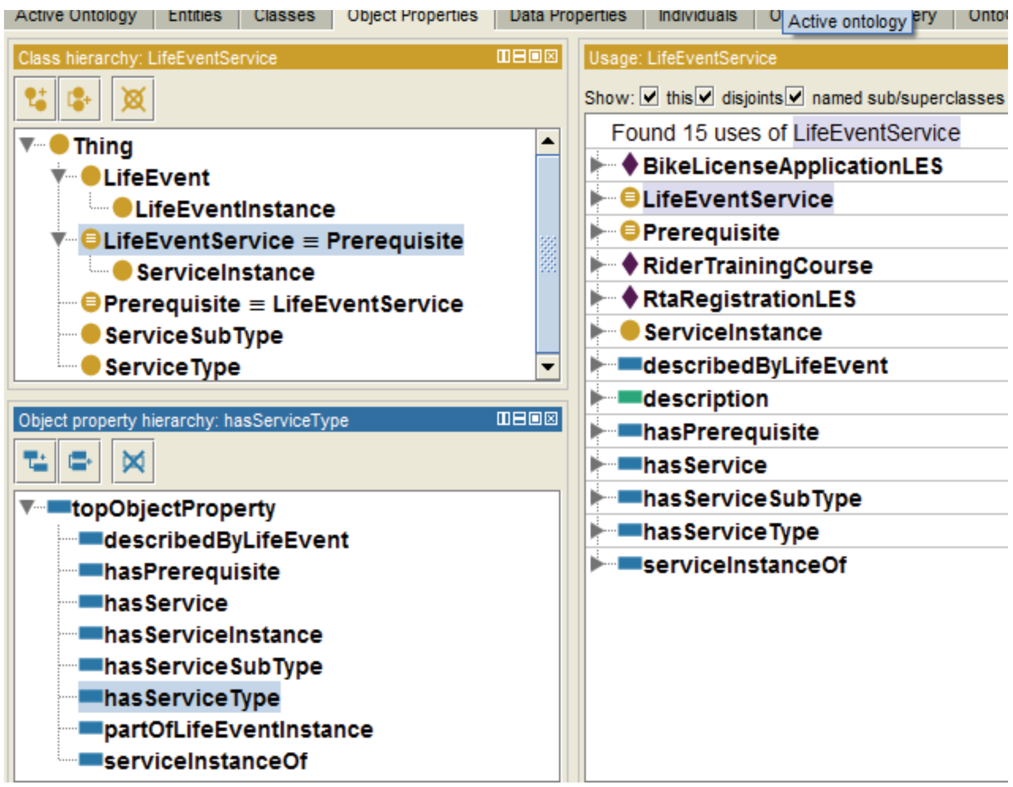
Where $C \in N'$ and symbol # denotes the cardinality and the E' denotes the set of entities.

8.3 Experiment Preparation

This section will describe the preparation for a comparative evaluation of LifeEvent ontology against OWL-S using the methods described in Sections 8.2.2 and 8.2.3.

8.3.1 Step 1: Building the Ontology

This step aims to prepare for the evaluation of the LifeEvent ontology. The choice of ontology editor was made mainly due to the fact that Protégé is open source software, and was more suited to our purpose (Stanford-University, 2009). This tool is developed and maintained by Stanford University. We used version 4.1, which was more advanced, intuitive and easier to use than other available ontology editors. Figure 8.1 is a screenshot of our developed ontology which illustrates the concepts, their relationships with object properties and data properties in the Protégé ontology editor.



Class hierarchy: LifeEventService

```
Thing
  LifeEvent
    LifeEventInstance
  LifeEventService ≡ Prerequisite
    ServiceInstance
  Prerequisite ≡ LifeEventService
  ServiceSubType
  ServiceType
```

Object property hierarchy: hasServiceType

```
topObjectProperty
  describedByLifeEvent
  hasPrerequisite
  hasService
  hasServiceInstance
  hasServiceSubType
  hasServiceType
  partOfLifeEventInstance
  serviceInstanceOf
```

Usage: LifeEventService

```
Found 15 uses of LifeEventService
  BikeLicenseApplicationLES
  LifeEventService
  Prerequisite
  RiderTrainingCourse
  RtaRegistrationLES
  ServiceInstance
  describedByLifeEvent
  description
  hasPrerequisite
  hasService
  hasServiceSubType
  hasServiceType
  serviceInstanceOf
```

Figure 8.1: LifeEvent Ontology Built by Protégé.

8.3.2 Step 2: Selecting a Comparable Ontology

This step selects an ontology that is conceptually similar to LifeEvent ontology. OWL-S is chosen because it is not only conceptually very similar to the LifeEvent ontology but also functionally designed to perform a similar task. This ontology is also used by the prototype to provide knowledgebase support for the web service enactment functionality of the system. The diagram in Figure 8.2 is the graph representation of OWL-S conceptual schema version 1.1.

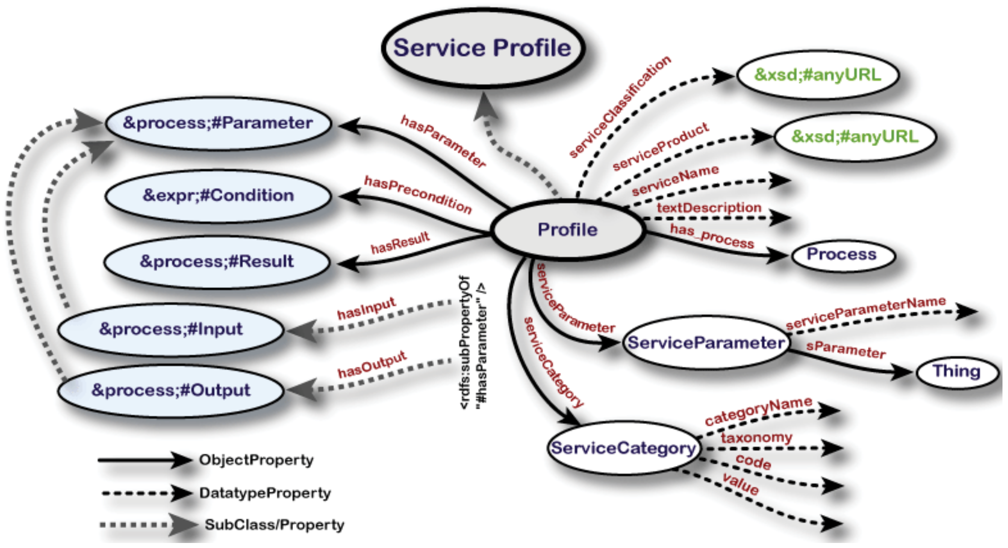
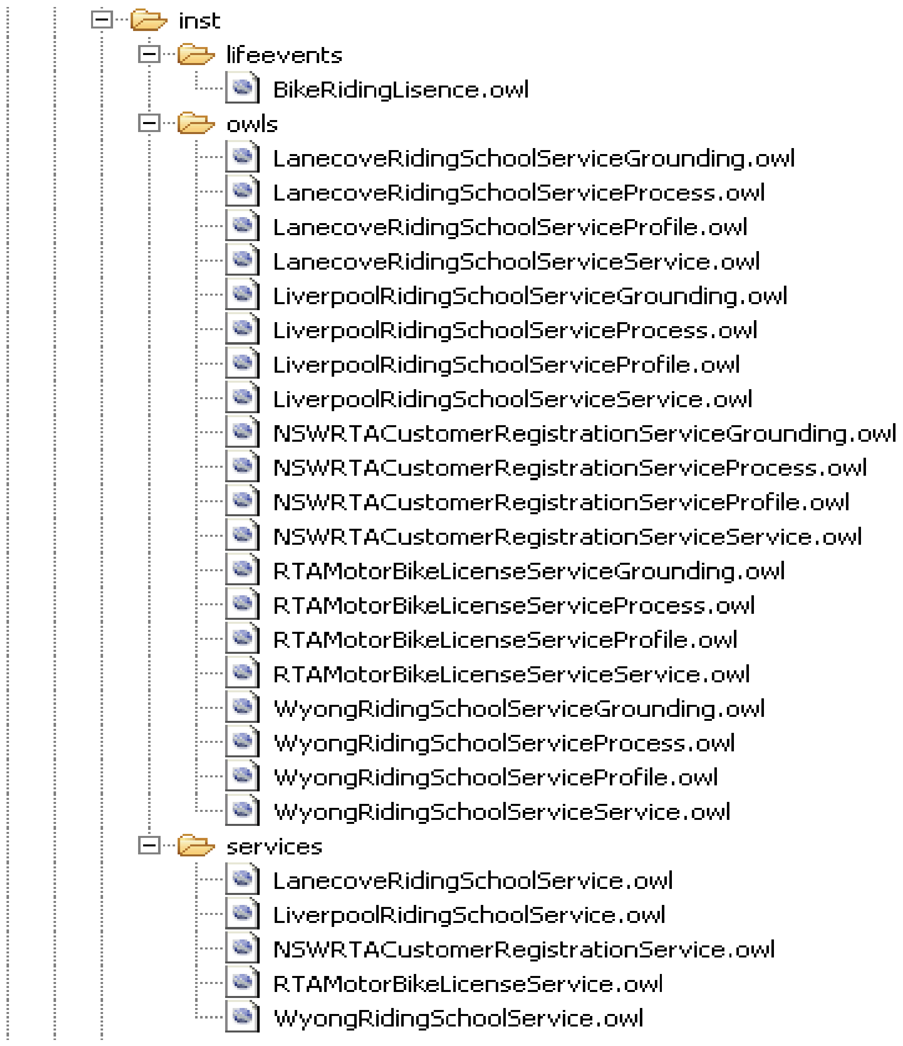


Figure 8.2: OWL-S Ontology schema (W3C, 2004)

In this comparative evaluation of LifeEvent ontology with OWL-S we use numerical value results by applying the metrics in Sections 8.2.2 and 8.2.3 on both ontologies to illustrate the measurement of efficiency and complexity of the LifeEvent ontology in comparison to the OWL-S.

8.3.3 Step 3: Experimentation

The experiment starts by creating and adding five named individuals that are the representatives of five individual web services that are published by the Australian Government agencies and other businesses. One more named individual is created only in the LifeEvent ontology as the first instance of the schema to point to the LifeEventServices. The list of these named individuals is described in Figure 8.3.



```
inst
  lifeevents
    BikeRidingLisence.owl
  owls
    LanecoveRidingSchoolServiceGrounding.owl
    LanecoveRidingSchoolServiceProcess.owl
    LanecoveRidingSchoolServiceProfile.owl
    LanecoveRidingSchoolServiceService.owl
    LiverpoolRidingSchoolServiceGrounding.owl
    LiverpoolRidingSchoolServiceProcess.owl
    LiverpoolRidingSchoolServiceProfile.owl
    LiverpoolRidingSchoolServiceService.owl
    NSWRTACustomerRegistrationServiceGrounding.owl
    NSWRTACustomerRegistrationServiceProcess.owl
    NSWRTACustomerRegistrationServiceProfile.owl
    NSWRTACustomerRegistrationServiceService.owl
    RTAMotorBikeLicenseServiceGrounding.owl
    RTAMotorBikeLicenseServiceProcess.owl
    RTAMotorBikeLicenseServiceProfile.owl
    RTAMotorBikeLicenseServiceService.owl
    WyongRidingSchoolServiceGrounding.owl
    WyongRidingSchoolServiceProcess.owl
    WyongRidingSchoolServiceProfile.owl
    WyongRidingSchoolServiceService.owl
  services
    LanecoveRidingSchoolService.owl
    LiverpoolRidingSchoolService.owl
    NSWRTACustomerRegistrationService.owl
    RTAMotorBikeLicenseService.owl
    WyongRidingSchoolService.owl
```

Figure 8.3: LifeEvent and OWL-S named individuals

Considering the populated OWL-S ontology and the LifeEvent Ontology, we use the actual measurements with the methods described in Sections 8.2.2 and 8.2.3 to calculate the results as follows:

1) Based on the SOV method we measure the SOV value of the LifeEvent to be 7+8=15, and for OWL-S to be 12+9=21. This means

that if we consider the growth ratio for the LifeEvent as being the base 15/15=1 then this for the OWL-S would be 21/15= 1.4. Table 8.1 details the numerical representation of the growth of ontology data in both ontologies.

We register the statistics in Table 8.1 by assuming the initial SOV to be the sum of the nodes, plus object properties in the ontology schema. We then increased this number five times as per the number of named individuals representing the web services created for this experiment, each time by the amount of SOV ratio, representing the linear growth in the volume of ontology data.

LifeEvent	**OWL-S**
15	21
30	36
60	86.4
120	207.36
240	497.7

Table 8.1: Numerical representation of ontology growth as per SOV ratio.

Figure 8.4 is the comparative graph representation that illustrates the trend of growth in the LifeEvent ontology data and the OWL-S ontology data. It is shown that the rate of growth in the volume of data in the LifeEvent is dramatically less than the OWL-S, after a fivefold increases in the number of named individuals.

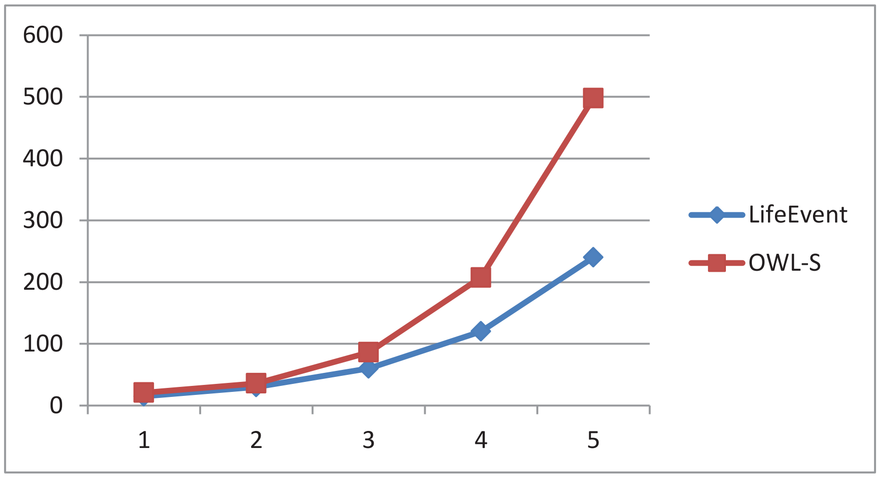


Figure 8.4: Physical representation of growths in ontology data as per SOV ratio

2) Based on the ENR method we measure the ENR value of both ontologies to be as follows:

LifeEvent= $\frac{|15|}{|7|} = 2.5$, OWL-S $\frac{|21|}{|8|} = 2.63$.

Table 8.2 shows the growth of ontology data in both ontologies in terms of ENR in numerical terms. The statistics in Table 8.2 are obtained by increasing the initial ENR by five times as per the number of web service named individuals, created for this experiment, each time by the amount of the ENR ratio.

LifeEvent	OWL-S
2.5	2.63
6.25	6.9
15.63	18.2
39.1	47.8
97.75	125.8

Table 8.2: Numerical representation of physical growth for ontology as per ENR ratio

Figure 8.5 is the comparative graph illustrating the trend of growth in ENR for the LifeEvent ontology and OWL-S ontology in the event of growth in ontology data. It is shown that this increase in LifeEvent Ontology is less than OWL-S after a few fold increases in the number of named individuals.

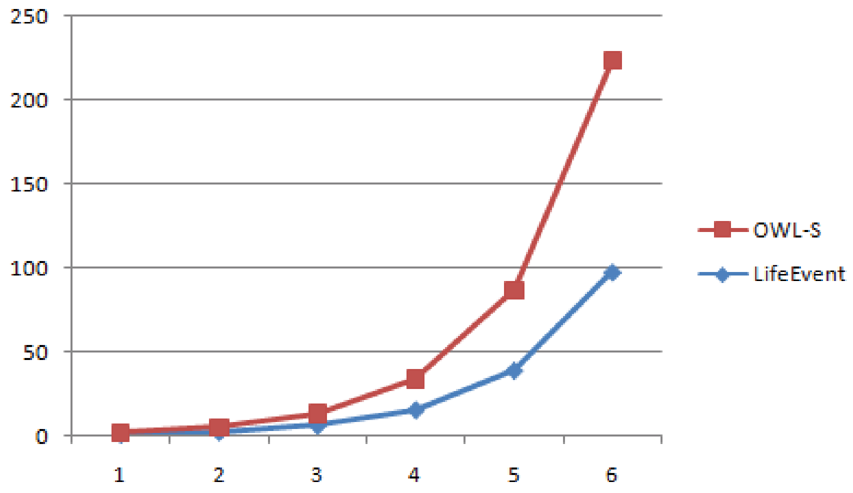


Figure 8.5: Physical representation of growths in ontology data per ENR ratio

3) Based on the TIP method we measure the value of 'how far LifeEvent deviates from being a tree?' to be (15-7+1=9), and for

OWL-S to be (21-8+1=14). It is shown this value is greater for OWL-S than for LifeEvent ontology.

4) Using the NOC method, we calculated the number of immediate children (*rdf: subClassOf*) for the class *Parameter* that is the most frequently used in web service invocation to be three. The value of NOC calculated for LifeEventService, which is the most used class in LifeEvent ontology is two. Table 8.3 shows the complexity growth of ontology data for a class Parameter in OWL-S in comparison with the class LifeEventService in LifeEvent ontology in terms of the NOC ratio. The statistics in Table 8.3 have been obtained by initial NOC increased five times as per the number of web service named individuals, created for this experiment, each time by the amount of NOC ratio.

LifeEvent	**OWL-S**
2	3
4	9
8	27
16	81
32	243

Table 8.3: Numerical representation of physical growth for ontology classes as per NOC ratio

Figure 8.6 is the comparative graph illustrating the trend of growth in complexity as per the NOC ratio for class Parameter in OWL-S in comparison with the class LifeEventService in LifeEvent ontology in the event of growth in ontology data. It is shown that this increase in

LifeEvent Ontology is less than OWL-S after a few fold increases in the number of named individuals.

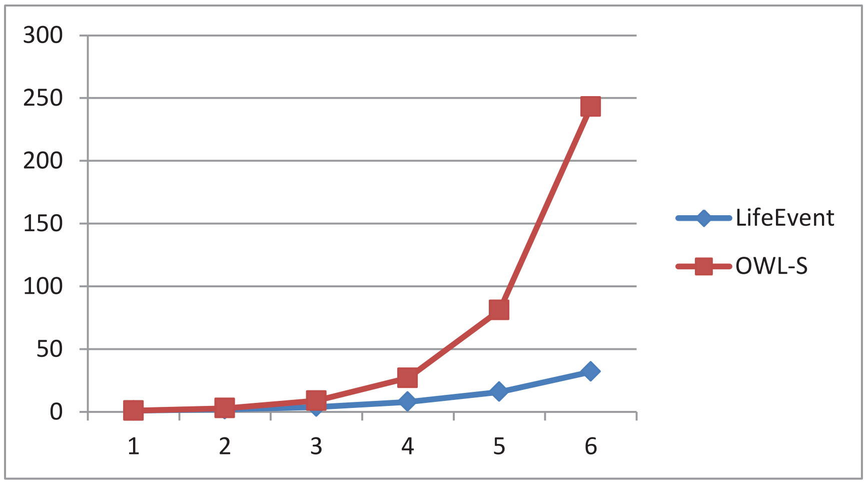


Figure 8.6: Physical representation of growths in complexity of ontology data as per NOC ratio

8.4 LOOSI Platform Prototype Evaluation

This section gives detailed description of our experimental run through the LOOSI application prototype.

8.4.1 Experiment Preparation

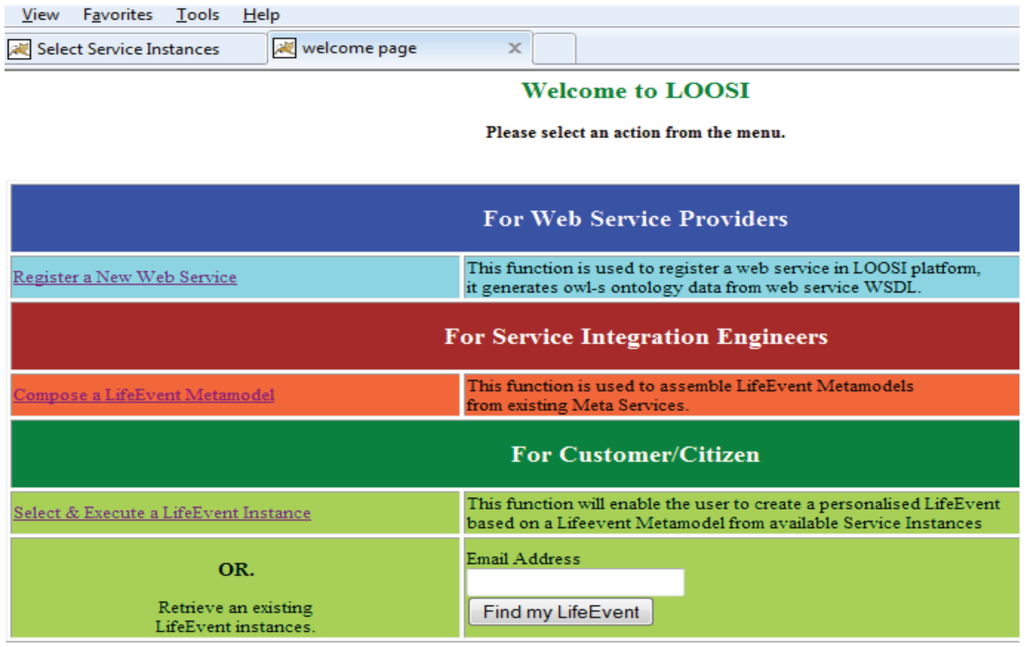
For the purpose of this experiment we already developed six web services and deployed them in to an AXIS2 server, these available web services are listed in Table 8.4.

#	Service Name	WSDL URL
1	NSWRTACustomerRegistrationService	http://localhost:8080/axis2/services/NSWRTACustomerRegistrationService?wsdl
2	RTAMotorBikeLicenseService	http://localhost:8080/axis2/services/RTAMotorBikeLicenseService?wsdl
3	RydeRidingSchoolService	http://localhost:8080/axis2/services/RydeRidingSchoolService?wsdl
4	LanecoveRidingSchoolService	http://localhost:8080/axis2/services/LanecoveRidingSchoolService?wsdl
5	WyongRidingSchoolService	http://localhost:8080/axis2/services/WyongRidingSchoolService?wsdl
6	LiverpoolRidingSchoolService	http://localhost:8080/axis2/services/LiverpoolRidingSchoolService?wsdl

Table 8.4: Readily available web service in AXIS2 test server

To start this experiment we go to the home page of the LOOSI project at *http://localhost:8080/LOOSIWeb/ DisplayMenu.action*. At this point we are presented with three functional options each of which related to one of the main stages of ESIM framework. Colour-coded scheme of the home page is also illustrating three distinct stages of ESIM framework (see Figure 8.7).

From here on we go through all three stages one by one and observe the comparison of the actual results with expected results. We consider activities in Sections 8.2.6 and 8.2.7 as to be the implementation of Stage 1 and 2 of ESIM framework as one project to develop a LifeEvent from a proposed candidate.



Welcome to LOOSI

Please select an action from the menu.

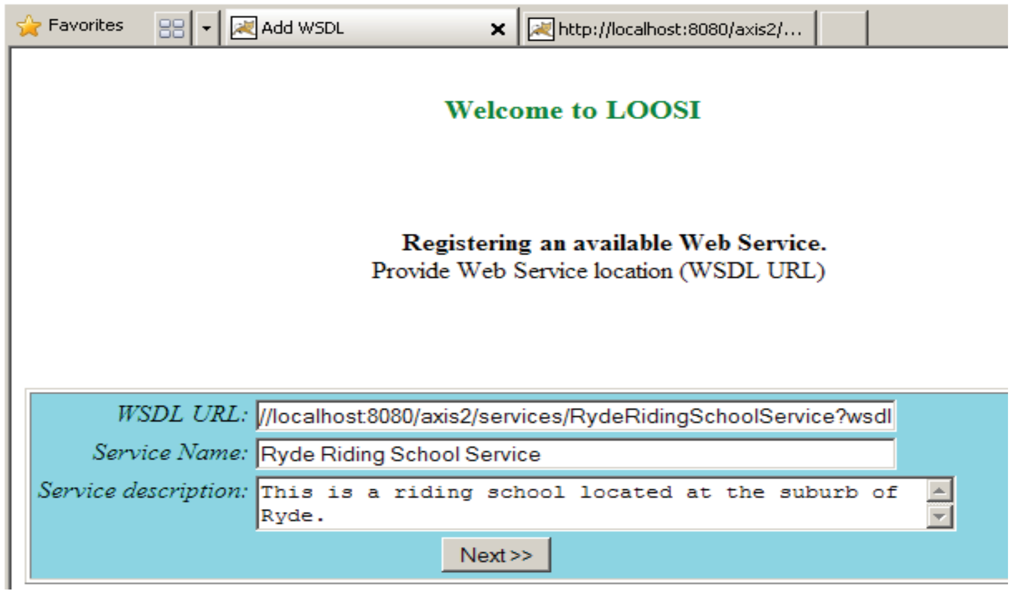
For Web Service Providers	
Register a New Web Service	This function is used to register a web service in LOOSI platform, it generates owl-s ontology data from web service WSDL.
For Service Integration Engineers	
Compose a LifeEvent Metamodel	This function is used to assemble LifeEvent Metamodels from existing Meta Services.
For Customer/Citizen	
Select & Execute a LifeEvent Instance	This function will enable the user to create a personalised LifeEvent based on a Lifeevent Metamodel from available Service Instances
OR. Retrieve an existing LifeEvent instances.	Email Address

Figure 8.7: Home page of LOOSI platform prototype

8.4.2 Register a New Web Service

By clicking on the link '*Register a New Web Service'* provided in the blue area of the home page, the user is directed to a page that enables the user to enter relevant syntactic and semantic information about the nominated web service. For the purpose of this experiment we choose to register the third service (RydeRidingSchoolService) from the list in Table 8.4.

The results of clicking on the first option are shown in Figure 8.8. Here the user is asked to enter the URL of the actual web service descriptor and some information about the service such as name and description. We enter the URL in Table 8.4 provided for the third web service and click 'Next'.



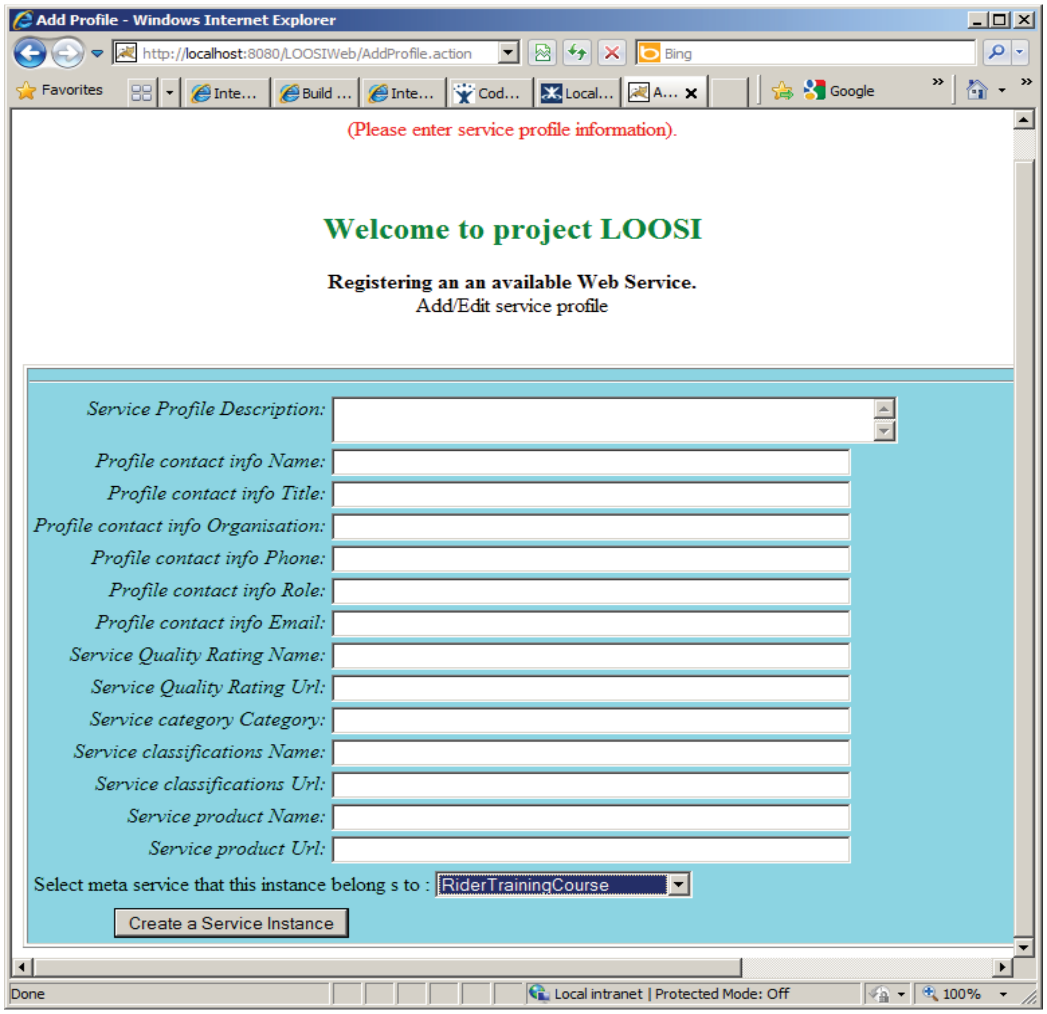
Welcome to LOOSI

Registering an available Web Service.
Provide Web Service location (WSDL URL)

WSDL URL:	//localhost:8080/axis2/services/RydeRidingSchoolService?wsdl
Service Name:	Ryde Riding School Service
Service description:	This is a riding school located at the suburb of Ryde.

Figure 8.8: Capturing the WSDL URL and service description

By clicking the 'Next' button we are directed to a page that provides input facility for semantic information required to build the OWL-S ontology data and 'LifeEvent Service Instance' data file. The image in Figure 8.9 shows the result of this action. Please note that at this point the user required to select an available service type in order to create an association with this service instance (we select the 'RiderTrainingCourse' for this service instance) and click "Create Service Instance'.


Add Profile - Windows Internet Explorer

http://localhost:8080/LOOSIWeb/AddProfile.action

(Please enter service profile information).

Welcome to project LOOSI

Registering an an available Web Service.
Add/Edit service profile

Service Profile Description:	
Profile contact info Name:	
Profile contact info Title:	
Profile contact info Organisation:	
Profile contact info Phone:	
Profile contact info Role:	
Profile contact info Email:	
Service Quality Rating Name:	
Service Quality Rating Url:	
Service category Category:	
Service classifications Name:	
Service classifications Url:	
Service product Name:	
Service product Url:	

Select meta service that this instance belong s to : RiderTrainingCourse

Create a Service Instance

Done | Local intranet | Protected Mode: Off | 100%

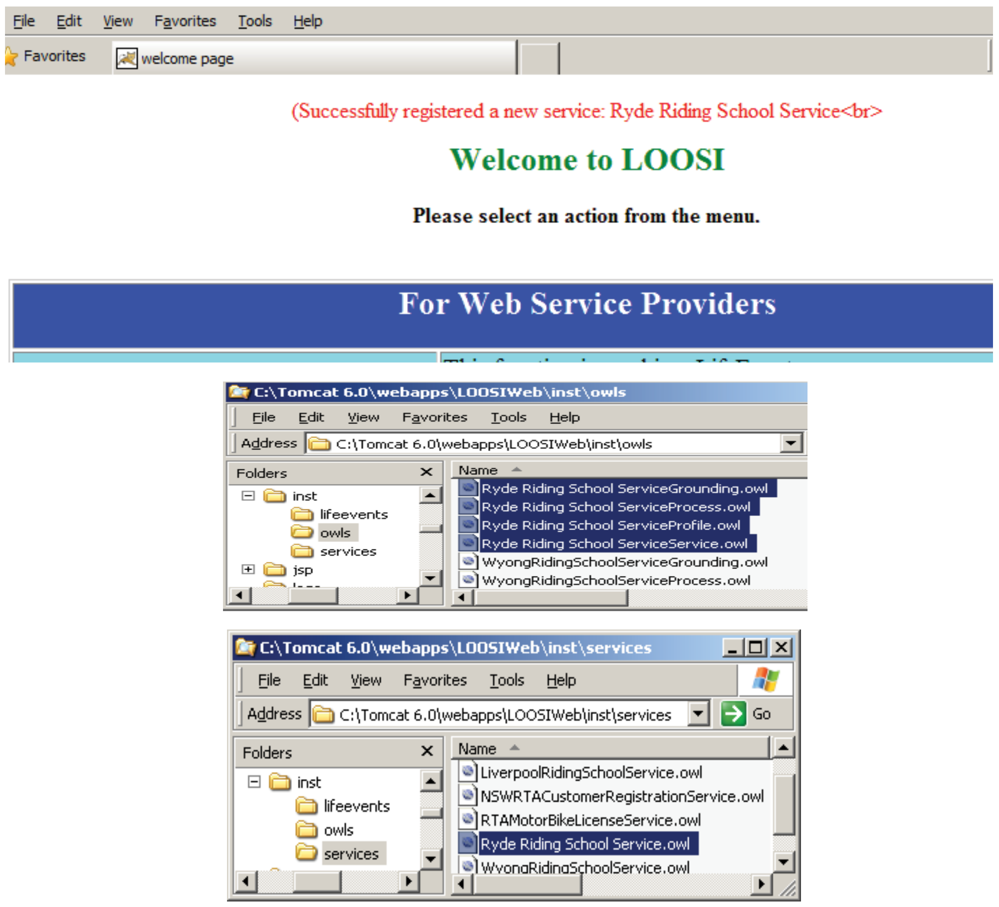
Figure 8.9: User semantic input to create the OWL-S ontology data

By clicking on the 'Create Service Instance' the system will perform the following actions:

1. Validate the existence of the WSDL URL and the availability of the web service
2. Created individuals for Grounding, Process, Profile and Service concepts in OWL-S ontology schema

3. Created an individual for ServiceInstance concept in LifeEvent ontology schema.

As a result of our last action the application has performed two main tasks associated with the ESIM framework: First created a complete set of OWL-S ontology data and second created a LifeEvent service instance ontology data file. Actual physical file results of these tasks are shown in Figure 8.10.


File Edit View Favorites Tools Help
Favorites
welcome page

(Successfully registered a new service: Ryde Riding School Service

Welcome to LOOSI

Please select an action from the menu.

For Web Service Providers

C:\Tomcat 6.0\webapps\LOOSIWeb\inst\owls
File Edit View Favorites Tools Help
Address C:\Tomcat 6.0\webapps\LOOSIWeb\inst\owls
Folders
inst
lifeevents
owls
services
jsp
Name

Ryde Riding School ServiceGrounding.owl
Ryde Riding School ServiceProcess.owl
Ryde Riding School ServiceProfile.owl
Ryde Riding School ServiceService.owl
WyongRidingSchoolServiceGrounding.owl
WyongRidingSchoolServiceProcess.owl

C:\Tomcat 6.0\webapps\LOOSIWeb\inst\services
File Edit View Favorites Tools Help
Address C:\Tomcat 6.0\webapps\LOOSIWeb\inst\services
Go
Folders
inst
lifeevents
owls
services
Name

LiverpoolRidingSchoolService.owl
NSWRTACustomerRegistrationService.owl
RTAMotorBikeLicenseService.owl
Ryde Riding School Service.owl
WyongRidingSchoolService.owl

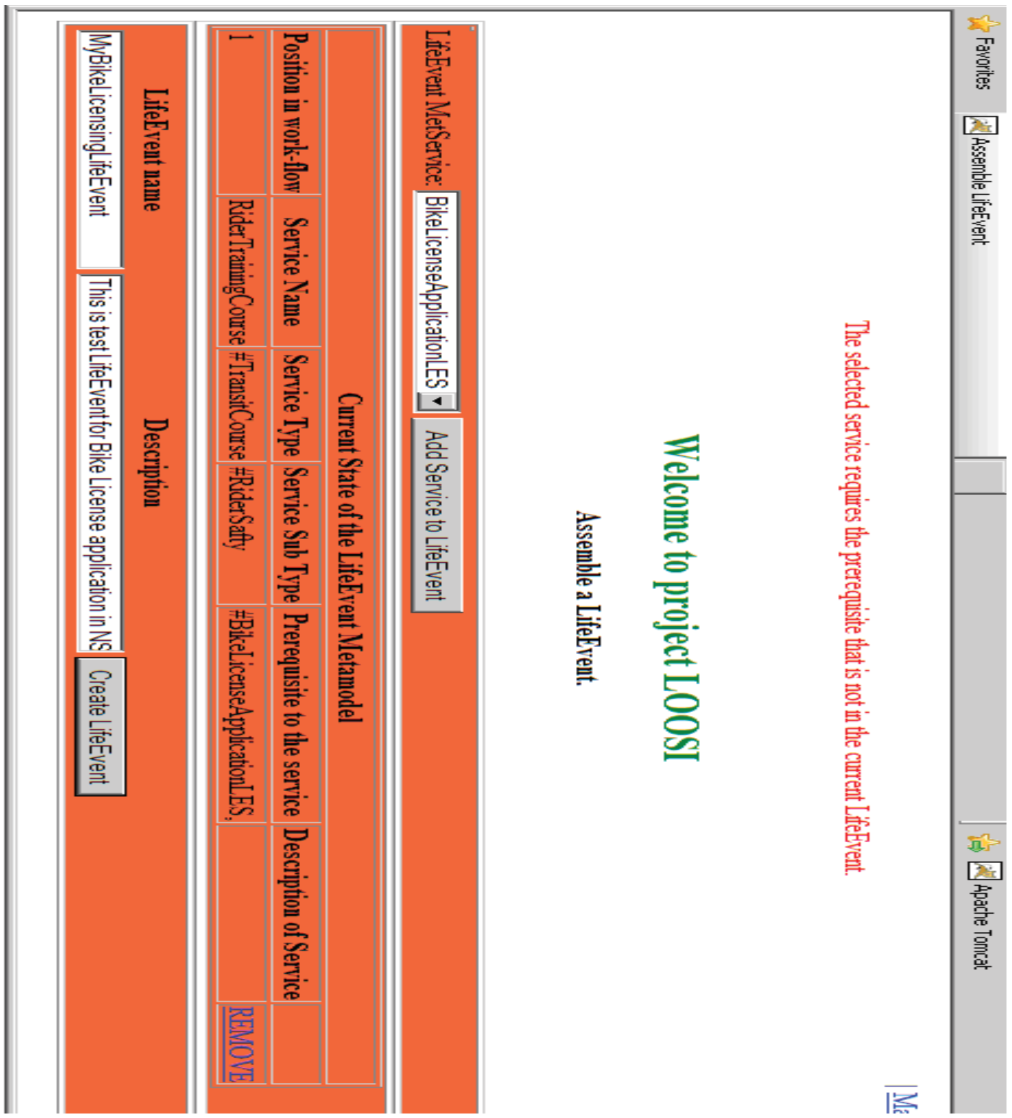
Figure 8.10: Actual results of executing the stage 1 of the ESIM by the prototype

8.4.3 Compose a LifeEvent Metamodel

According to the second stage of the ESIM framework integration engineers are able to create LifeEvent Metamodels. This function is provided by prototype through clicking on the link '*Compose a LifeEvent Metamodel'* provided in the Red area of the home page, Integration engineers are directed to a page that enables the user to assemble a LifeEvent Metamodel. In the page 'Assemble a LifeEvent' user enters the name and the description of the candidate Metamodel that is to be created, then the user can add available MetaServices to the workflow of the LifeEvent. We note that the system is enforcing the regulatory rules to assemble the workflow. this is done by blocking the action of adding the MetaServices in a wrong order from the e-government regulations prospective. This regulatory knowledge is provided by LifeEvent ontology at run time to ensure a legally acceptable outcome for the execution of the LifeEvent. For example if we try to create a LifeEvent for Rider Licensing then the work flow of MetaServices must be in the following order:

1) RtaRegistrationLES
2) BikeLicenseApplicationLES
3) RiderTrainingCourse

Any other order will generate an error message. To test this function first we entered 'RiderTrainingCourse' as the first MetaService, second when tried to enter 'RtaRegistrationLES', we receive an error message indicating that the prerequisite of the first MetaService (BikeLicenseApplicationLES) is not satisfied. This function and the error message are illustrated in Figure 8.11.



The selected service requires the prerequisite that is not in the current LifeEvent.

Welcome to project LOOSI

Assemble a LifeEvent.

LifeEvent MetService: BikeLicenseApplicationLES

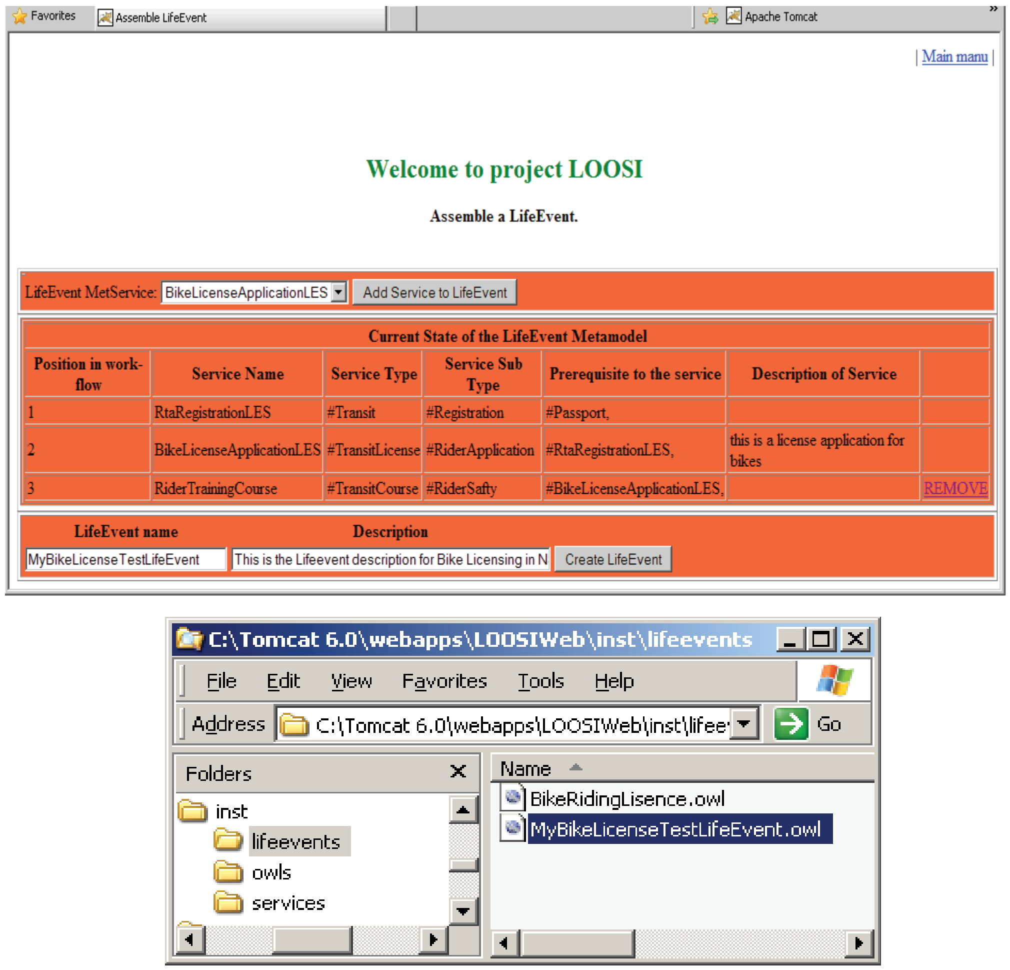
Current State of the LifeEvent Metamodel

Position in work-flow	Service Name	Service Type	Service Sub Type	Prerequisite to the service	Description of Service	
1	RiderTrainingCourse	#TransitCourse	#RiderSafty	#BikeLicenseApplicationLES,		REMOVE

LifeEvent name	Description	
MyBikeLicensingLifeEvent	This is test LifeEvent for Bike License application in NS	Create LifeEvent

Figure 8.11: Enforcing the regulatory knowledge provided by LifeEvent ontology

As soon as we remove the service from the position and replace it with the correct one the system accepts the LifeEvent and allows the user to proceed with the LifeEvent Metamodel assembly (see Figure 8.12).



Welcome to project LOOSI

Assemble a LifeEvent.

LifeEvent MetService: BikeLicenseApplicationLES

Current State of the LifeEvent Metamodel

Position in work-flow	Service Name	Service Type	Service Sub Type	Prerequisite to the service	Description of Service	
1	RtaRegistrationLES	#Transit	#Registration	#Passport,		
2	BikeLicenseApplicationLES	#TransitLicense	#RiderApplication	#RtaRegistrationLES,	this is a license application for bikes	
3	RiderTrainingCourse	#TransitCourse	#RiderSafty	#BikeLicenseApplicationLES,		REMOVE

LifeEvent name	Description	
MyBikeLicenseTestLifeEvent	This is the Lifeevent description for Bike Licensing in N	Create LifeEvent

C:\Tomcat 6.0\webapps\LOOSIWeb\inst\lifeevents

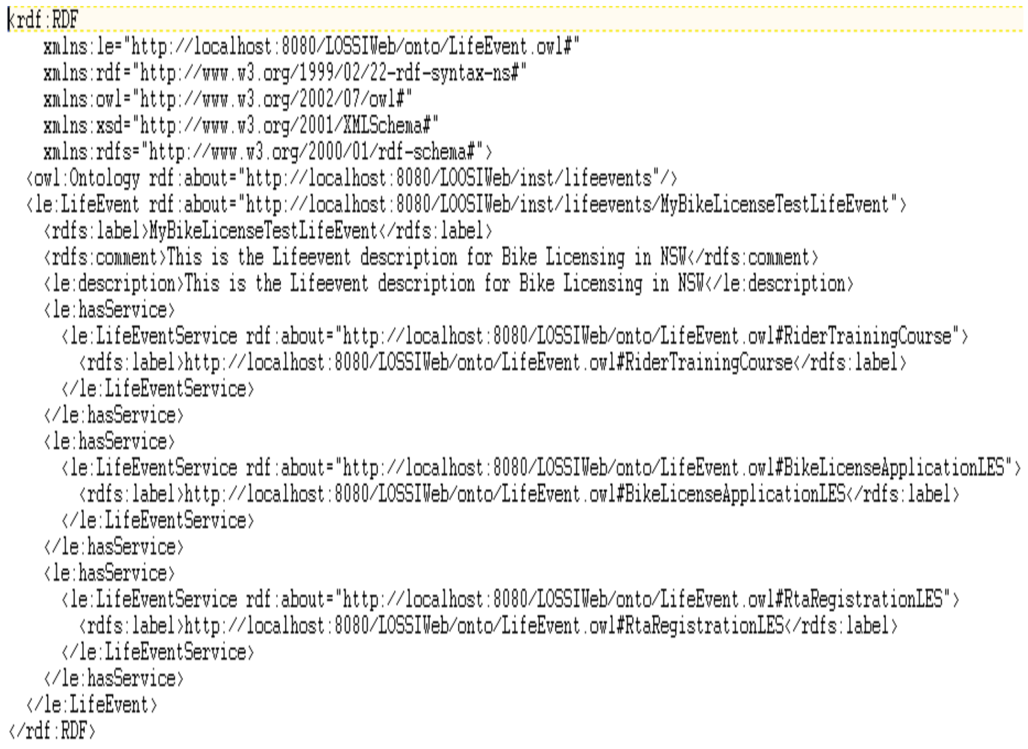
Address: C:\Tomcat 6.0\webapps\LOOSIWeb\inst\lifee

inst
- lifeevents
- owls
- services

Name
- BikeRidingLisence.owl
- MyBikeLicenseTestLifeEvent.owl

Figure 8.12: Successful assembly of a LifeEvent Metamodel

This function generates the ontology data file shown in Figure 8.12. We inspect this file closer and see all the RDF statements in the ontology data. Apart from data properties (Label, comment and description), we can see three object properties of predicates ‘*hasService’* is constructed pointing to the three MetaServices selected by the user through the execution of this Stage 2 of ESIM framework (see Listing 8.1).



```
<rdf:RDF
    xmlns:le="http://localhost:8080/LOSSIWeb/onto/LifeEvent.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <owl:Ontology rdf:about="http://localhost:8080/LOOSIWeb/inst/lifeevents"/>
  <le:LifeEvent rdf:about="http://localhost:8080/LOOSIWeb/inst/lifeevents/MyBikeLicenseTestLifeEvent">
    <rdfs:label>MyBikeLicenseTestLifeEvent</rdfs:label>
    <rdfs:comment>This is the Lifeevent description for Bike Licensing in NSW</rdfs:comment>
    <le:description>This is the Lifeevent description for Bike Licensing in NSW</le:description>
    <le:hasService>
      <le:LifeEventService rdf:about="http://localhost:8080/LOSSIWeb/onto/LifeEvent.owl#RiderTrainingCourse">
        <rdfs:label>http://localhost:8080/LOSSIWeb/onto/LifeEvent.owl#RiderTrainingCourse</rdfs:label>
      </le:LifeEventService>
    </le:hasService>
    <le:hasService>
      <le:LifeEventService rdf:about="http://localhost:8080/LOSSIWeb/onto/LifeEvent.owl#BikeLicenseApplicationLES">
        <rdfs:label>http://localhost:8080/LOSSIWeb/onto/LifeEvent.owl#BikeLicenseApplicationLES</rdfs:label>
      </le:LifeEventService>
    </le:hasService>
    <le:hasService>
      <le:LifeEventService rdf:about="http://localhost:8080/LOSSIWeb/onto/LifeEvent.owl#RtaRegistrationLES">
        <rdfs:label>http://localhost:8080/LOSSIWeb/onto/LifeEvent.owl#RtaRegistrationLES</rdfs:label>
      </le:LifeEventService>
    </le:hasService>
  </le:LifeEvent>
</rdf:RDF>
```

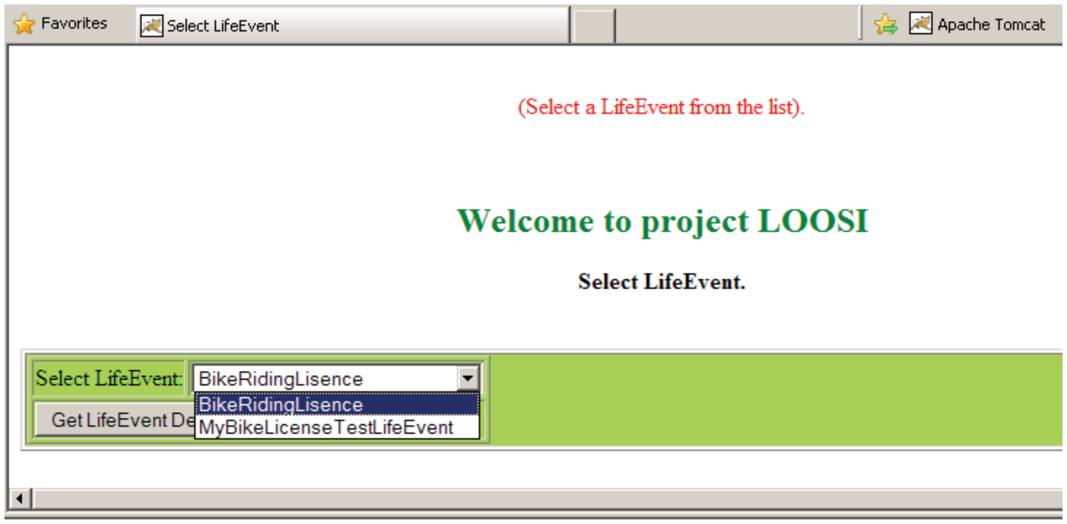
Listing 8.1: RDF statements in LifeEvent ontology data file

This action marks the end of Stage 2 of ESIM framework in this project. Now the system has stored enough information and knowledge to offer one LifeEvent Metamodel, so ordinary citizens are able to customise and execute their personalised instance of the LifeEvent.

8.4.4 Select and Execute a LifeEvent Instance

According to the second stage of the ESIM framework service consumers are able to use the system to select a LifeEvent Metamodel, create a personalised instance and execute the instance in a user controlled environment. We initiate this function by clicking on the link (Select & Execute a LifeEvent Instance) provided in the main page of the prototype.

This action takes the user to a page that presents a list of available LifeEvent Metamodels to choose from (see Figure 8.13). Here we can see our newly created LifeEvent Metamodel along with an existing one.



(Select a LifeEvent from the list).

Welcome to project LOOSI

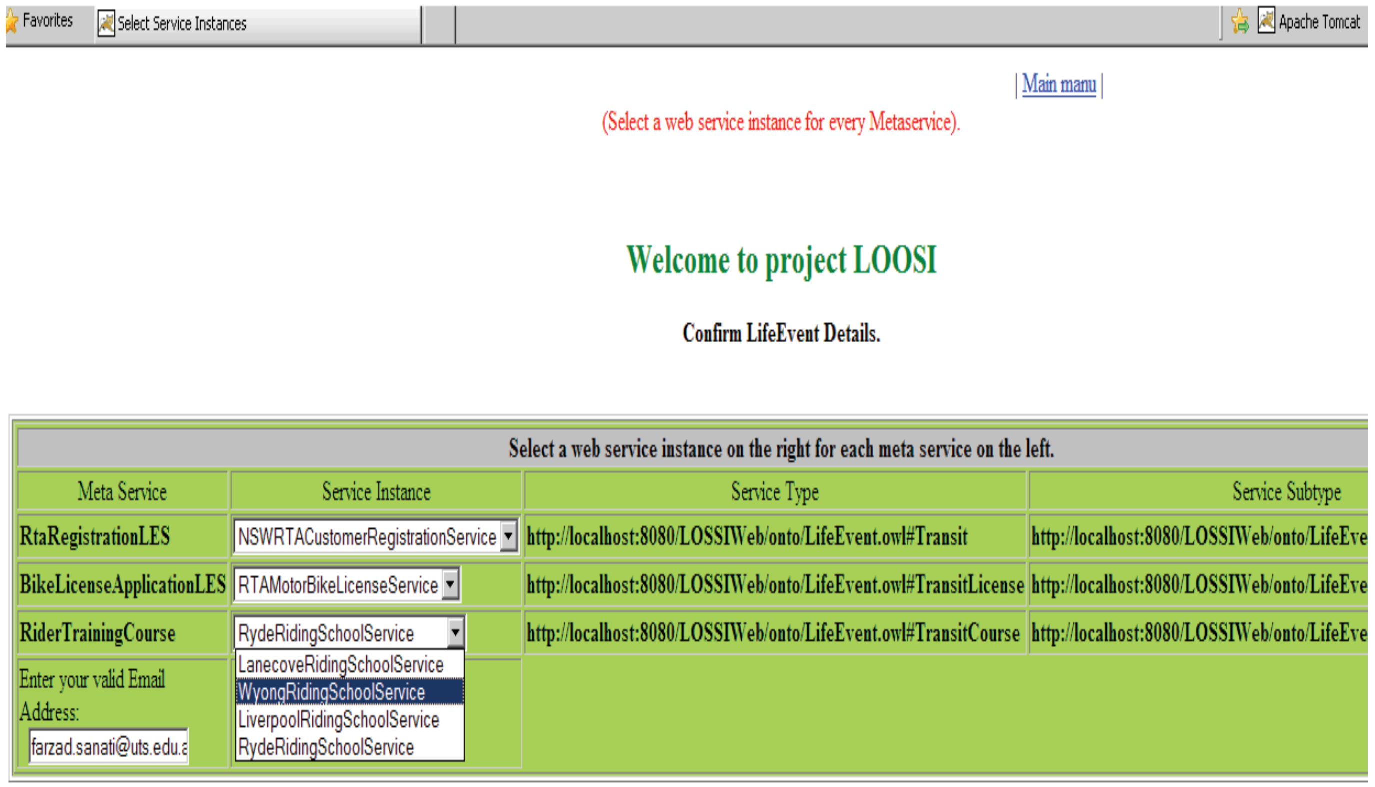
Select LifeEvent.

Select LifeEvent: BikeRidingLisence

BikeRidingLisence
MyBikeLicenseTestLifeEvent

Figure 8.13: Selecting LifeEvent Metamodel

We select the one we just created and click on the ‘Get LifeEvent Details’ button. This takes us to a new page that displays the details of the Metamodel and allows user to create personalise instance of the Metamodel by substituting desired web service instances in the place of every MetaService in the LifeEvent. We customise our LifeEvent Instance, enter an email address as customer identification for future references and click on ‘Create LifeEvent Instance’ (see Figure 8.14).



(Select a web service instance for every Metaservice).

Welcome to project LOOSI

Confirm LifeEvent Details.

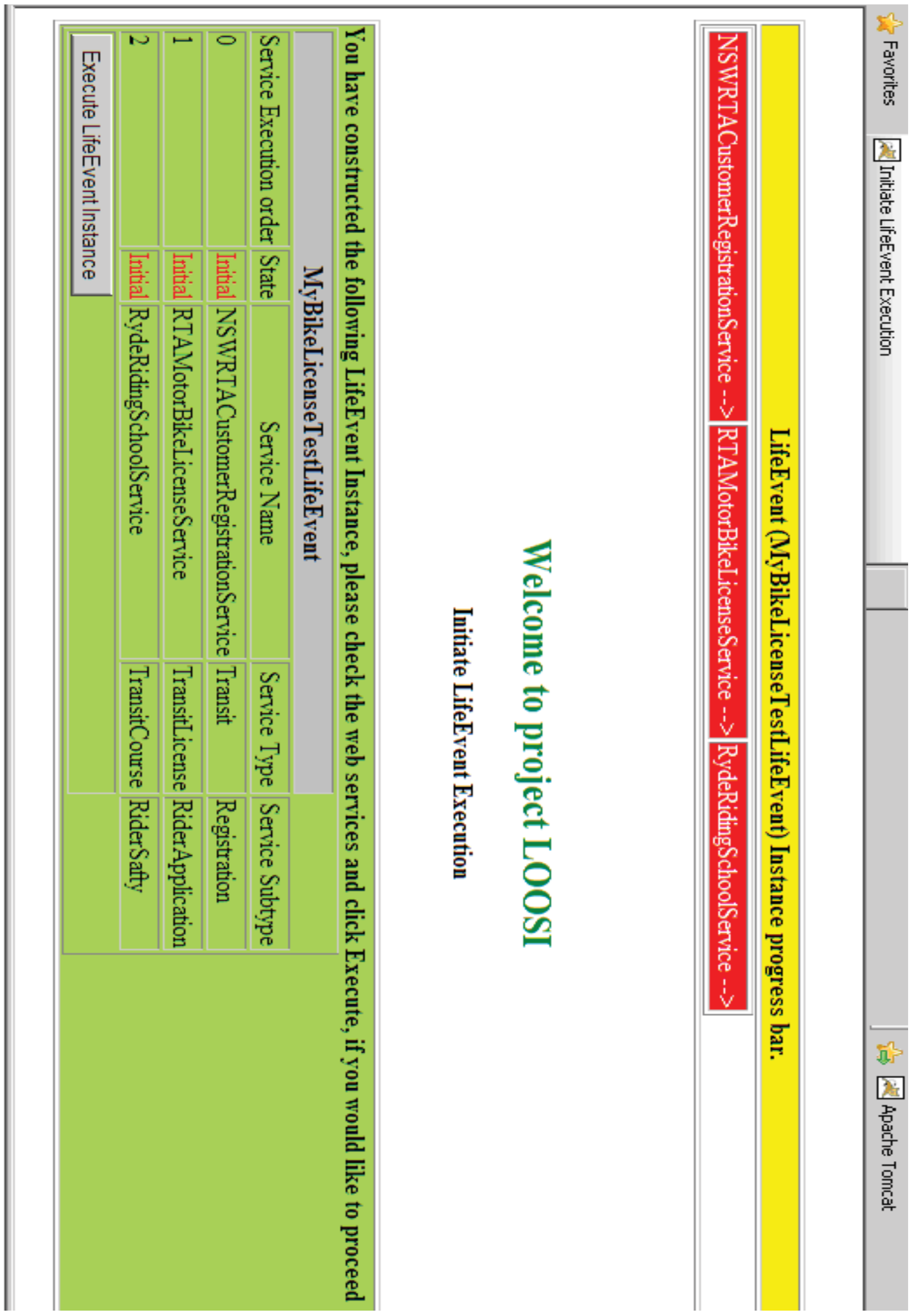
Select a web service instance on the right for each meta service on the left.

Meta Service	Service Instance	Service Type	Service Subtype
RtaRegistrationLES	NSWRTACustomerRegistrationService	http://localhost:8080/LOSSIWeb/onto/LifeEvent.owl#Transit	http://localhost:8080/LOSSIWeb/onto/LifeEve
BikeLicenseApplicationLES	RTAMotorBikeLicenseService	http://localhost:8080/LOSSIWeb/onto/LifeEvent.owl#TransitLicense	http://localhost:8080/LOSSIWeb/onto/LifeEve
RiderTrainingCourse	RydeRidingSchoolService	http://localhost:8080/LOSSIWeb/onto/LifeEvent.owl#TransitCourse	http://localhost:8080/LOSSIWeb/onto/LifeEve
Enter your valid Email Address: farzad.sanati@uts.edu.a	LanecoveRidingSchoolService WyongRidingSchoolService LiverpoolRidingSchoolService RydeRidingSchoolService		

Figure 8.14: Personalising LifeEvent Metamodel

Next page displays the details of newly created personalised LifeEvent instance to confirm the initiation and the execution of the instance by clicking on the ‘Execute LifeEvent Instance’.

The system also provides a progress bar at the top of the page that indicates the direction and the progress of the current execution of individual web services. The instance is initially created on the memory, and creates persistence for future references and long running transitions when user confirms the initiation of the LifeEvent Instance (see Figure 8.15).



LifeEvent (MyBikeLicenseTestLifeEvent) Instance progress bar.

NSWRTACustomerRegistrationService --> RTAMotorBikeLicenseService --> RydeRidingSchoolService -->

Welcome to project LOOSI

Initiate LifeEvent Execution

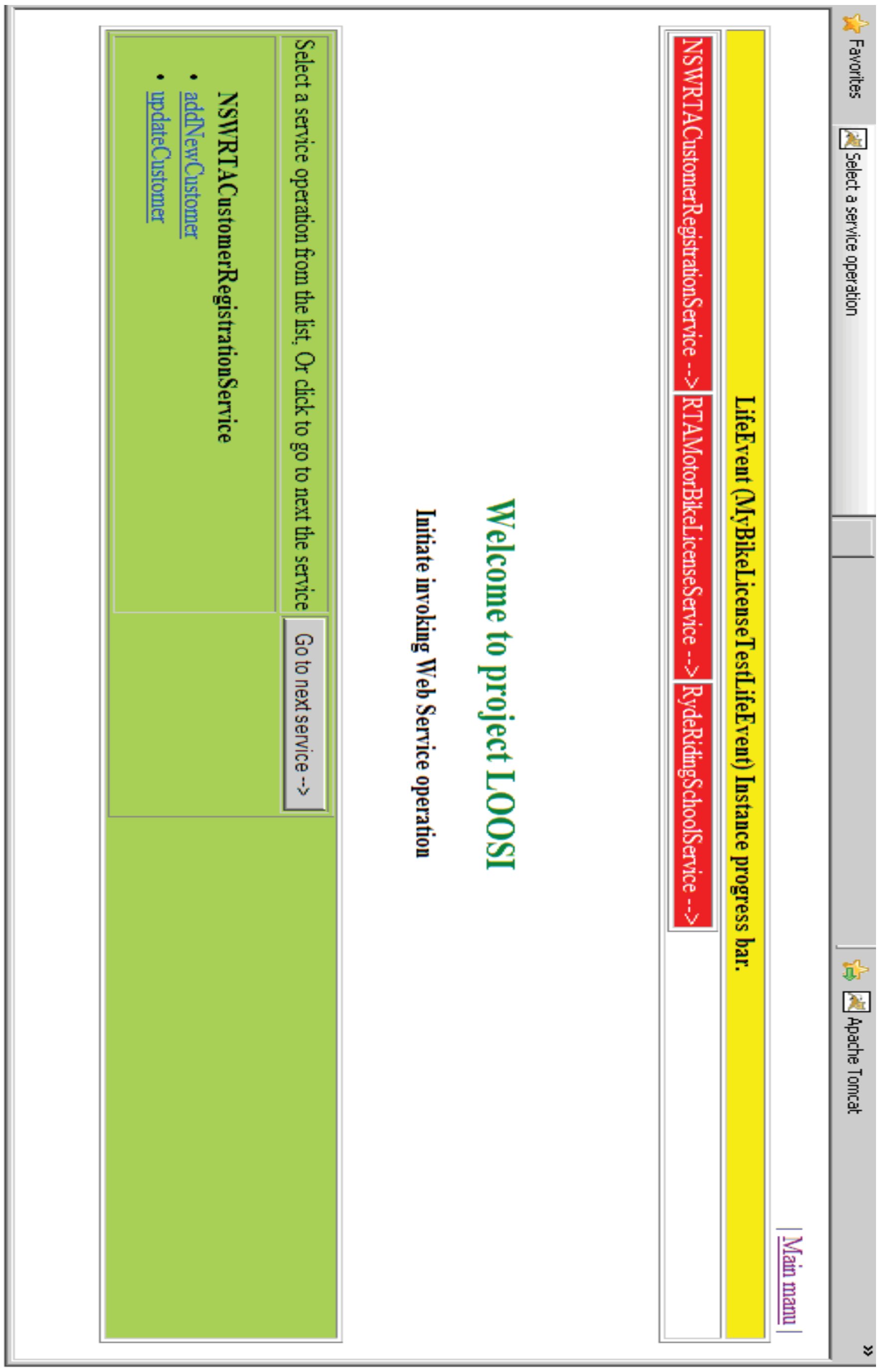
You have constructed the following LifeEvent Instance, please check the web services and click Execute, if you would like to proceed

MyBikeLicenseTestLifeEvent

Service Execution order	State	Service Name	Service Type	Service Subtype
0	Initial	NSWRTACustomerRegistrationService	Transit	Registration
1	Initial	RTAMotorBikeLicenseService	TransitLicense	RiderApplication
2	Initial	RydeRidingSchoolService	TransitCourse	RiderSafty

Figure 8.15: Confirming the execution of personalised LifeEvent instance

Upon clicking on the ‘Execute LifeEvent Instance’ system loads all the available semantic information from both OWL-S and LifeEvent ontology data files, and displays all the operations available in the first web service for user to select and execute. Alternatively user can choose to skip the operation of the current web service and go to the next service by clicking on the ‘Go to Next Service’ (see Figure 8.16).


Favorites | Select a service operation | Apache Tomcat

| Main manu |

LifeEvent (MyBikeLicenseTestLifeEvent) Instance progress bar.

NSWRTACustomerRegistrationService --> | RTAMotorBikeLicenseService --> | RydeRidingSchoolService -->

Welcome to project LOOSI

Initiate invoking Web Service operation

Select a service operation from the list, Or click to go to next the service

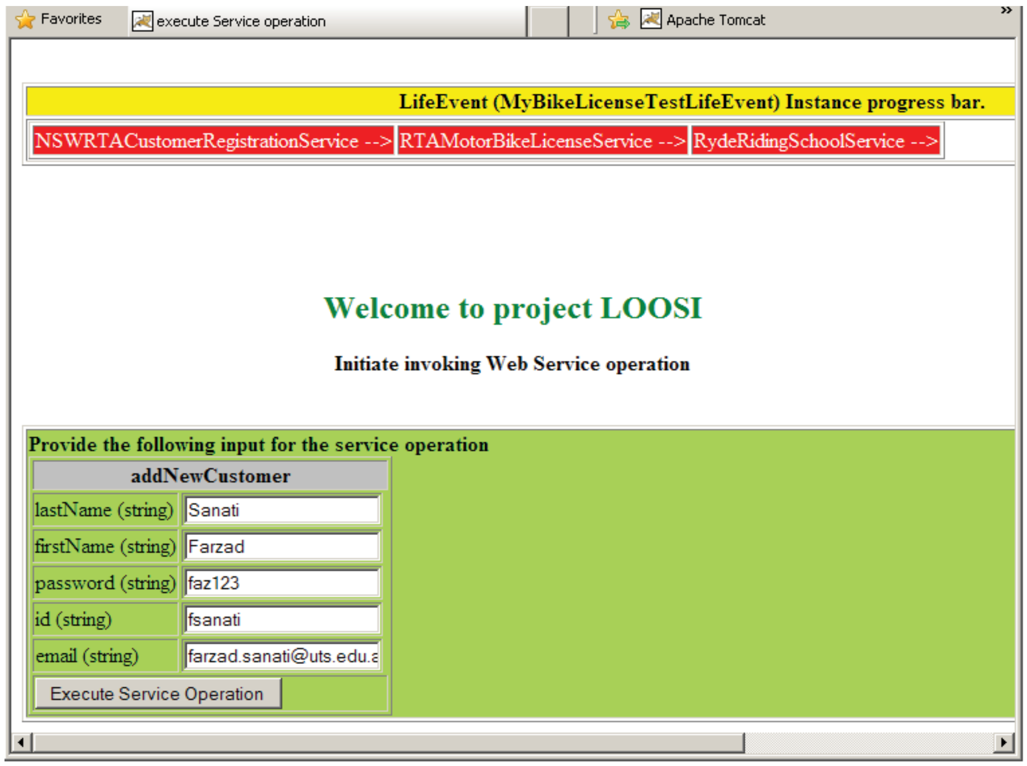
Go to next service -->

NSWRTACustomerRegistrationService

- addNewCustomer
- updateCustomer

Figure 8.16: Select an operation from the currently web service

In this time we select the operation ‘addNewCustomer’ from the current executing service (NSWRTA CustomerRegistrationService) in the workflow. When an operation is selected, the system interrogates the OWL-S ontology data files to prepare an input screen. This system uses the ‘Semantic Ontology Manager’ component to retrieve the information and then uses the ‘Unified Presentation & User Interface’ component to interpret and dynamically render the appropriate input screen for users to enter their data in order to communicate with the service provider (see Figure 8.17).



LifeEvent (MyBikeLicenseTestLifeEvent) Instance progress bar.

NSWRTACustomerRegistrationService --> RTAMotorBikeLicenseService --> RydeRidingSchoolService -->

Welcome to project LOOSI

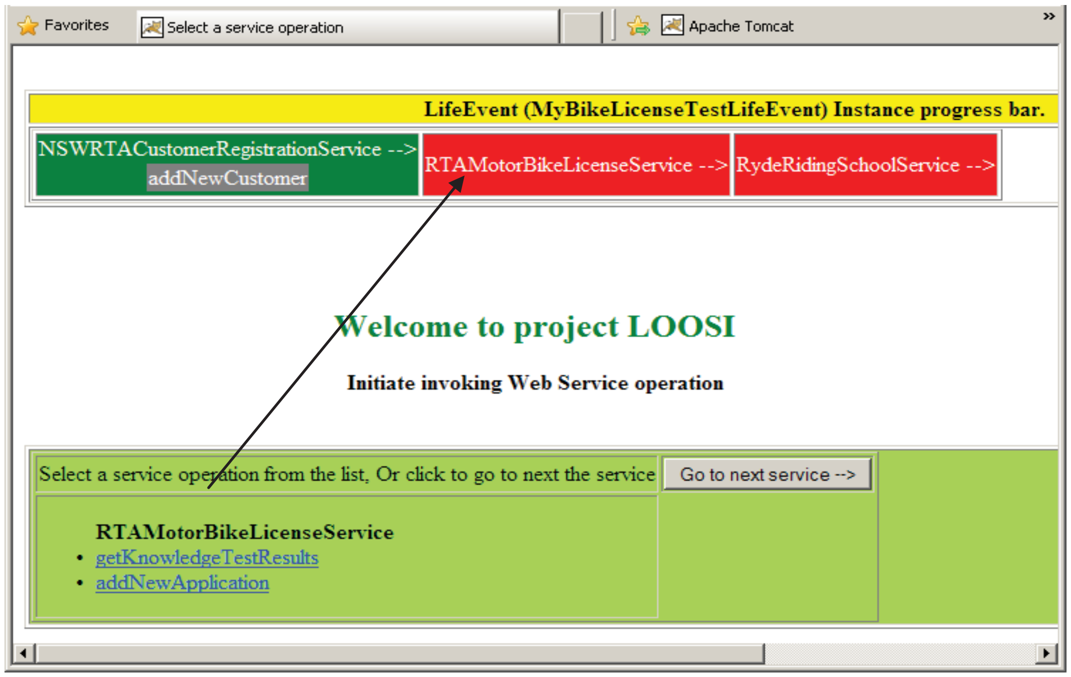
Initiate invoking Web Service operation

Provide the following input for the service operation

addNewCustomer	
lastName (string)	Sanati
firstName (string)	Farzad
password (string)	faz123
id (string)	fsanati
email (string)	farzad.sanati@uts.edu.a

Figure 8.17: User data entry screen to invoke an operation from the currently web service

Clicking on the ‘Execute Service Operation’ will result on invocation of the selected operation from the current service and system will display any values returned by the remote web service if the invocation is successful. In the next screen the name of the invoked operation will be displayed in gray colour under the name of the current service indicative of being already invoked. The user can stay in this screen and invoke different services from the current service until ‘Go to Next Service’ is clicked. Then the progress bar at the top of the screen will show this service as green (meaning that the user is done with this service) and change the currently executing service to the next one (RTA Motor Bike License Service in our case). And similar to the last service all the available operations of this system is listed under the name of the service (see Figure 8.18).



LifeEvent (MyBikeLicenseTestLifeEvent) Instance progress bar.

NSWRTACustomerRegistrationService --> addNewCustomer

RTAMotorBikeLicenseService -->

RydeRidingSchoolService -->

Welcome to project LOOSI

Initiate invoking Web Service operation

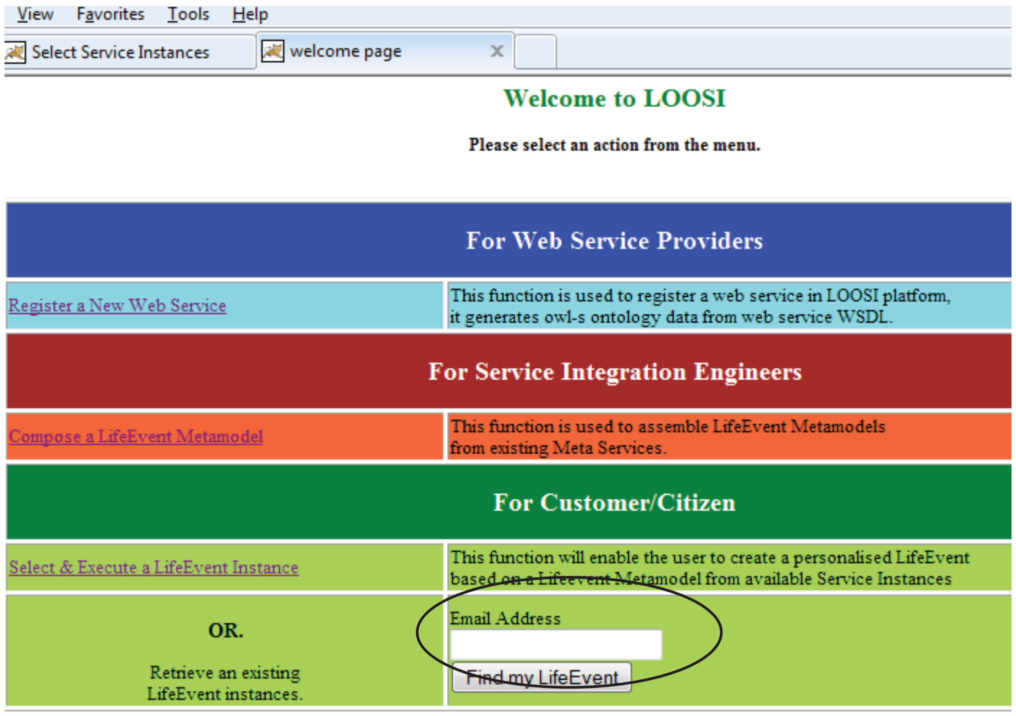
Select a service operation from the list, Or click to go to next the service

RTAMotorBikeLicenseService

- getKnowledgeTestResults
- addNewApplication

Figure 8.18: Finalised web services are coloured as green on the progress bar.

User can continue this until all services are invoked and all operations are executed. Alternatively user can choose to leave the system and come back later (due to possible long running transitions or personal reason). The system caters for this function via saving the current state of the LifeEvent Instance at the end of every operation so the user can come back later and continue the LifeEvent Instance by providing a user id this is illustrated in Figures 8.19 and 8.20.

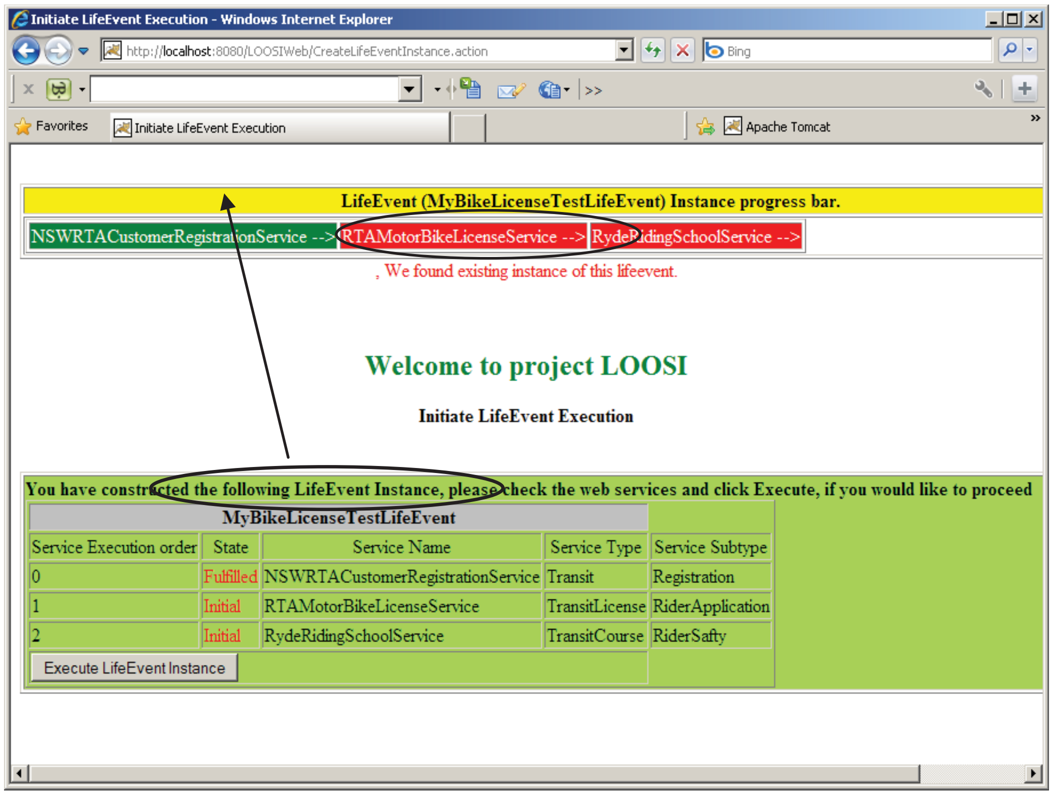


Welcome to LOOSI

Please select an action from the menu.

For Web Service Providers	
Register a New Web Service	This function is used to register a web service in LOOSI platform, it generates owl-s ontology data from web service WSDL.
For Service Integration Engineers	
Compose a LifeEvent Metamodel	This function is used to assemble LifeEvent Metamodels from existing Meta Services.
For Customer/Citizen	
Select & Execute a LifeEvent Instance	This function will enable the user to create a personalised LifeEvent based ~~on a Lifeevent Metamodel~~ from available Service Instances
OR. Retrieve an existing LifeEvent instances.	Email Address

Figure 8.19: Retrieving an incomplete LifeEvent Instance


Initiate LifeEvent Execution - Windows Internet Explorer
http://localhost:8080/LOOSIWeb/CreateLifeEventInstance.action
Favorites
Initiate LifeEvent Execution
Apache Tomcat
Execute LifeEvent Instance

LifeEvent (MyBikeLicenseTestLifeEvent) Instance progress bar.

NSWRTACustomerRegistrationService --> RTAMotorBikeLicenseService --> RydeRidingSchoolService -->

, We found existing instance of this lifeevent.

Welcome to project LOOSI

Initiate LifeEvent Execution

You have constructed the following LifeEvent Instance, please check the web services and click Execute, if you would like to proceed

MyBikeLicenseTestLifeEvent

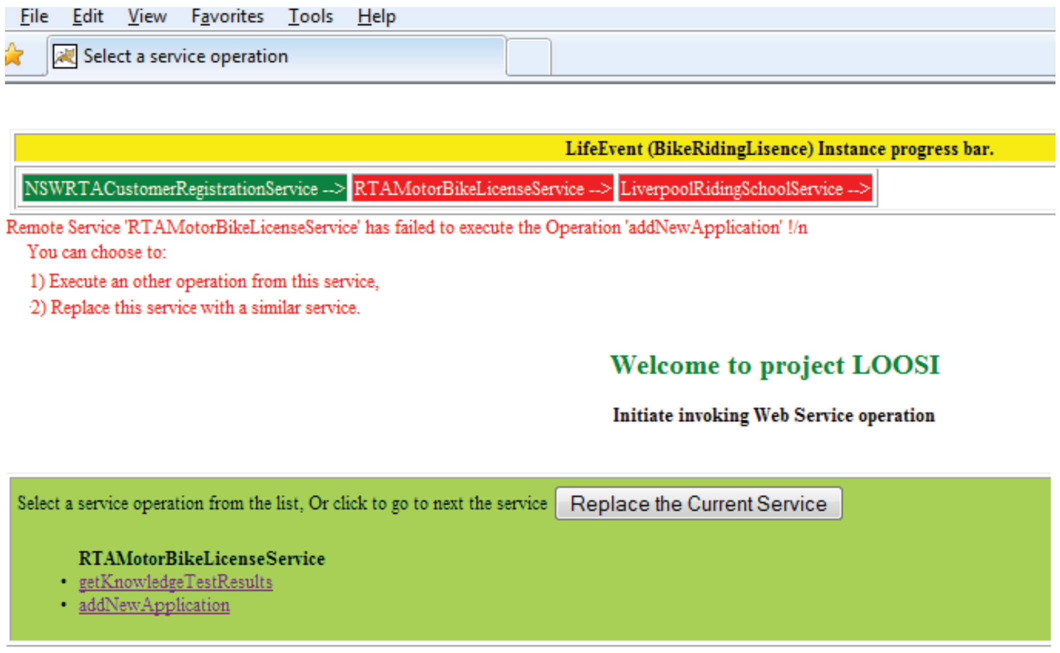
Service Execution order	State	Service Name	Service Type	Service Subtype
0	Fulfilled	NSWRTACustomerRegistrationService	Transit	Registration
1	Initial	RTAMotorBikeLicenseService	TransitLicense	RiderApplication
2	Initial	RydeRidingSchoolService	TransitCourse	RiderSafty

Figure 8.20: Revisiting an incomplete LifeEvent Instance.

It is possible that some time due to unforeseen circumstances one of the web services in the LifeEvent workflow fail to respond to the invocation call from the system. In such situation the system must be able to repair the LifeEvent workflow by replacing the failed web service with an alternative similar web service, available from the repository.

We tested this functionality by shutting down one of the web services in the LifeEvent workflow to investigate the results. In this case we shut down the web service 'RTAMotorBikeLicenseService' in the second in the position of the LifeEvent Instance and tried to continue the execution of the LifeEvent. The aforementioned action had the following results:

The system conveys a message to the user stating the situation; this message is illustrated in Figure 8.21.


File
Edit
View
Favorites
Tools
Help
Select a service operation
LifeEvent (BikeRidingLisence) Instance progress bar.
NSWRTACustomerRegistrationService -->
RTAMotorBikeLicenseService -->
LiverpoolRidingSchoolService -->
Replace the Current Service

Remote Service 'RTAMotorBikeLicenseService' has failed to execute the Operation 'addNewApplication' !/n
You can choose to:
1) Execute an other operation from this service,
2) Replace this service with a similar service.

Welcome to project LOOSI

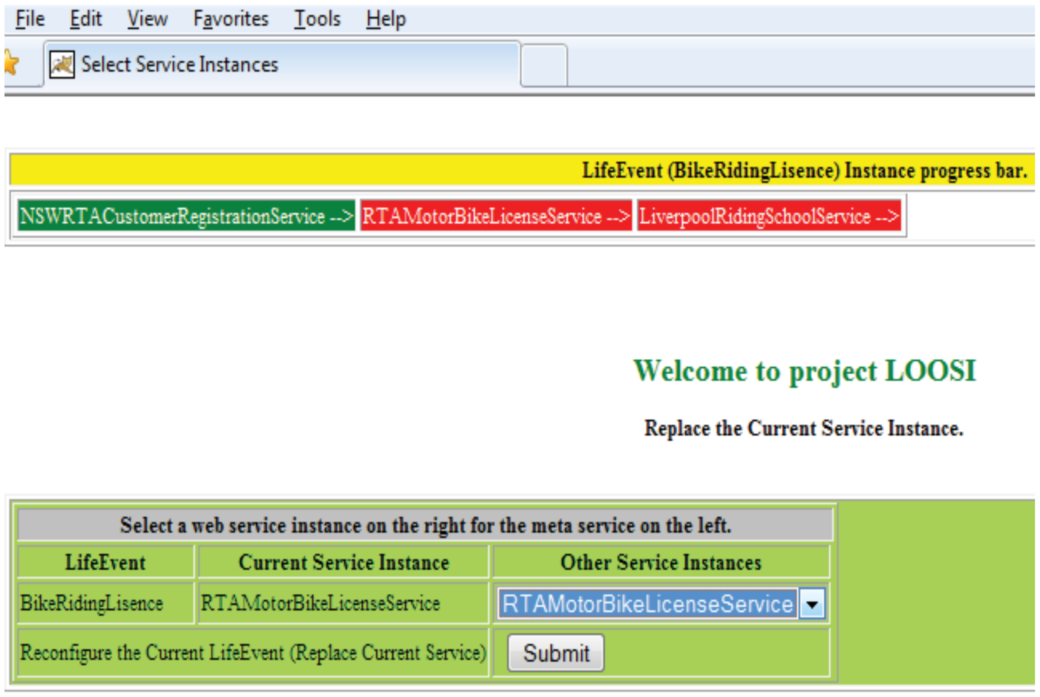
Initiate invoking Web Service operation

Select a service operation from the list, Or click to go to next the service

RTAMotorBikeLicenseService
- getKnowledgeTestResults
- addNewApplication

Figure 8.21: Web service failure notification

By clicking on the available button ‘Replace the Current Service’ we are directed to a page that provides the information about the current failed service and offers a list of alternative similar web services to replace the failed one. Figure 8.22 illustrates the results of this action.



LifeEvent (BikeRidingLisence) Instance progress bar.

NSWRTACustomerRegistrationService --> RTAMotorBikeLicenseService --> LiverpoolRidingSchoolService -->

Welcome to project LOOSI

Replace the Current Service Instance.

Select a web service instance on the right for the meta service on the left.		
LifeEvent	Current Service Instance	Other Service Instances
BikeRidingLisence	RTAMotorBikeLicenseService	RTAMotorBikeLicenseService
Reconfigure the Current LifeEvent (Replace Current Service)		Submit

Figure 8.22: LifeEvent repair action

We acknowledge that the LOOSI application prototype may have some negligible defects due to incorrect user data entry, lack of security implementation, and commercial software quality assurances since it is not a full-scale commercial application. However this experimental implementation using the available ontology data has confirmed all our formal design assumptions made in Chapters 3, 4, 5, 6 and 7 to be true.

Chapter 9: Conclusion and Future Work

9.1 Problem Area and objectives

Through literature review to various aspects of the e-government service integration and delivery, we recognised the major areas of knowledge that was of concern for evaluating the state of research in e-government.

This study recognised that the various approaches to e-government development model lacking one or another aspect of the reality. Argument was made that a more comprehensive model is needed to reflect better on the reality of e-government evolutionary transformation trend.

Type, State and the Maturity of technology to support the e-government integration was another area investigated by this research. The majority of recent research agrees that the web service and SOA are the most appropriate and popular integration technologies among other technologies such as CORBA. We also closely looked at the mechanism of support for intelligent architecture that could deliver a dynamically assembled and configured composite web services. OWL-S and WSMO have had the most influence in this area of research out of many other initiatives. We highlighted three major shortcomings of these technologies that need to be compensated if a truly dynamic integration of e-government services was to be achieved.

Broad knowledge was required about a model that could encapsulate the requirements of service integration (a unit of requirement) to achieve a dynamic integration of e-government services. We argued that the concept of LifeEvent is well suited to be used as the bases of such model.

We found four issues stoping the e-government service integration to become a success as follows:

1) Incompatible interagency process and workflows
2) Unfamiliar semantics
3) Complex inter-agency regulatory rules
4) Development of duplicated and redundant services
5) Lack of a comprehensive model for automatic composition and delivery of web services

Our analysis of these issues have lead us to pose two main questions of this research that would cover aforementioned issues, these questions are:

1) What is the most intelligent framework for e-government service integration and delivery?
2) What semantic web technologies can be used to overcome the technical difficulty of automatic integration and delivery of e-government service, and how?

To answer these research questions we needed to define clear objectives that could lead this research towards a final solution for the two questions above. Hence, the main objectives of this study are recognised as follows:

1) To investigate currently proposed e-government service integration models. This was essential to discover implementation and technical strategies of current solutions
2) To propose detailed specifications of a repeatable modelling framework for delivery of e-government integrated services.

3) To propose a delivery platform for integrated e-government services, this prototype application needs to make use of semantic web technology to create and manage personalised configuration for composite e-governments service workflows.

9.2 The Main Contributions of This Research

9.2.1 In Theory

This research proposed an implementation framework for e-government service integration and delivery. We anticipated that the findings of this research to have an important impact on efficiency and success of future e-government service integration and delivery projects by providing a unified methodological approach to such projects.

From engineering process point of view, to focus on the detail implementation of integrated e-government delivery system requires a specific attention to tasks and activities that are crucial to a successful service integration project. Hence we proposed and put on work a modified version of classical software engineering process. This modified process (SIE) has all the tasks and activities that are necessary to handle the complexity of service integration projects.

This research outlined the design concepts to extend the OWL-S in order to propose LifeEvent Ontology as a semantic knowledgebase for composing e-government web services. This ontology accommodates the concept of LifeEvent within the process of e-government web services integration. The idea used in this ontology design introduces an innovative approach towards the whole process of e-government web service integration and delivery. The design of LifeEvent ontology knowledgebase can manage the workflow of integrated web services.

However we also recognise that more research is required to specify and formalise the design of more complex types of web services composition that are not covered by this research, such as parallel service processes in complex workflows.

The concept of LifeEvent has been used to encapsulate the requirements of an integrated complex e-government service, composed from multiple web service, provided by possibly many service providers. The LifeEvent model is also designed for ordinary citizens to create and invoke their own personalised LifeEvent. For that to happen, LifeEvent had to be a Metamodel rather than an ordinary crystallised Model. This way it would be flexible enough to allow users to instantiate it in every possible way.

A modelling framework (ESIM) is proposed that uses the concept of LifeEvent and provides a process framework for the complete life cycle of LifeEvent from the design and creation of the Metamodel to the reconfiguration and instantiation of a LifeEvent instance by a service users.

9.2.2 In Practice

This research described the design and implementation of a novel online software system, used as an integration and delivery platform for integrated e-government services (LOOSI). This platform makes use of LifeEvent ontology as the knowledgebase to assist in decision making on dynamic integration and reconfiguration of LifeEvent services at run-time. LOOSI is designed based on the SOA architecture, it utilises the maximum power of using open source software applications and persistence. This software system is designed to adopt web-based enterprise application architecture based on MVC design pattern. LOOSI platform allows service providers to register their readily available web services. The result of this registration is a service ontology data file and an instance of a LifeEventService. Ontology data created during service registration provides the bases for the creation of a LifeEvent ontology that is a Metamodel form of LifeEvent. Finally all the ontology data produced during the service registration and Metamodel creation is used by ordinary citizens to create a personalised instance of LifeEvent and executed to solve a business problem.

9.3 Future Work

Although we have achieved a great successes in tackling the issues related to dynamic integration of e-government services and web services integration in general, there still exist more research tasks that are worth taking in the future work. In another sense, the issues in this research area are so significant and challenging and the topics are so attractive that a further pursuit is worthwhile. For example, intelligent transformation of government online services provides high quality information in an integrated communication environment; further research may be able to evolve this into a knowledge management system giving service across the entire government body. Nevertheless as a short reference, the future work is discussed in the following several subsections respectively based on the current studies discussed in this thesis.

1) ***Enhancements to the composition engine (Semantic Ontology Manager):*** This engine can be extended to support extended large integrated services. This will be particularly useful when service repositories grow extremely large in size due to integration with none-

government services. Also, the engine can be extended to find other kinds of compositions with loops such as repeat-until and iterations.

2) ***Web services integration in other industries:*** Composition of web service from multiple vendors can be applied to other industries such as finance and insurance. There is a great business opportunity for industries other than Public Service to take advantage of added value of the integrated web services to their customers. For example specific industries such as assurance may form a taskforce to develop a unifying service protocol and the required infrastructure to support and take advantage of the solution of this thesis. However in a technical note, this would require drastic changes in the control structure and extension of regulatory knowledgebase and some other areas of the original solution.
3) ***Service integration for embedded systems and Mobile-Computing:*** Web services can also be used in embedded systems and mobile devices which have limited processing power and memory. Accessing and using web services, web service selection, and integration in such devices will need light-weight and high-performance solutions.Adapting our current solutions for embedded systems and mobile devices will be considered as part of future work.
4) ***Type systems and type checking of Web services:*** The composition engine produces new functionality by joining or putting together, different web services. We also need to ensure the safe interoperability of these services involved in the composition. A typing system for the languages that describe Web services can help formalise the process of type verification and thereby ensuring safe interoperability.

ABRIVIATIONS

BPEL4WS	Business Process Engineering Language for Web Services
DAO	Data Access Object
DFD	Data Flow Diagram
e-GIF	e-Government Interoperability Framework
ER	Entity Relationship
ESIM	E-government Service Integration Modelling
FOL	First Order Logic
G2B	Government to Business
G2C	Government to Citizen
G2G	Government to Government
ICT	Information and Communication Technology
LE	LifeEvent
LEI	LifeEvent Instance
LEM	LifeEvent Metamodel
LES	LifeEvent Service
LOOSI	LifeEvent Ontology Oriented Integrated Services
MS	Meta Service
MVC	Model-View-Controller
NS	Name Space
OOD	Object Oriented Design
OWL	Web Ontology Language
OWL-S	Ontology Web Language for Services
OWL-GL	OWL Query Language
PR	Prerequisite
QoS	Quality of Service
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SAWSDL	Semantic Annotations for Web Service Description Language
SDLC	Software Development Life Cycle
SE	Software Engineering

SI	Service Instance
SIE	Service Integration Engineering
SQL	Simple Query Language
SOC	Service Oriented Computing
SOA	Service Oriented Architecture
SST	Service Sub Type
ST	Service Type
URI	Universal Resource Identifier
UTS	University of Technology Sydney
W3C	World Wide Web Consortium
WSDL	Web Service Description Language
WSMF	Web Service Modelling Framework
WSMO	Web Service Modelling Ontology
WSMX	Web Service modelling eXecution environment
XML	Extensible Markup Language

REFERENCES

AIS. 2007. *Design Research in Information Systems* [Online]. Association for Information Systems, http://home.aisnet.org/displaycommon.cfm?an=1&subarticlenbr=794#resourcesForDesignResearch. Available: http://home.aisnet.org/displaycommon.cfm?an=1&subarticlenbr=794#resourcesForDesignResearch [Accessed 30 December 2008 2008].

ANDERSON, P. 2006. *Impact of eGovernment on Territorial Government Service* [Online]. TERREGOV, http://www.terregov.eupm.net/my_spip/index.php. Available: http://www.terregov.eupm.net/my_spip/index.php [Accessed].

ANTHOPOULOS, L. G., SIOZOS, P. & TSOUKALAS, I. A. 2007. Applying participatory design and collaboration in digital public services for discovering and re-designing e-Government services. *Government Information Quarterly,* 24**,** 353-376.

APACHE. 2007. *Apache Struts 2 documentation* [Online]. http://struts.apache.org/2.x/index.html, accessed 24/04/2011. Available: http://struts.apache.org/2.x/index.html [Accessed 24/04/2011 2011].

ARROYO, S., SICILIA, M.-A. & DODERO, J.-M. 2006. Choreography frameworks for business integration: Addressing heterogeneous semantics. *Computers in Industry,* In Press, Corrected Proof.

AUSTASMANIAGOV. 2008. *Tasmania Life Event Services* [Online]. http://www.service.tas.gov.au/LifeEvents/. Available: http://www.service.tas.gov.au/LifeEvents/ [Accessed 15/10/2010].

AUSTRIAGOV. 2009. *HELP* [Online]. http://www.help.gv.at/. Available: http://www.help.gv.at/ [Accessed 9/5/2010].

AUSVICTORIAGOV. 2006. *Victoria E-government Initiatives* [Online]. http://www.egov.vic.gov.au/victorian-government-resources/government-2-0-action-plan.html. Available: http://www.egov.vic.gov.au/victorian-government-resources/government-2-0-action-plan.html [Accessed 15/10/2010].

BAEZA-YATES, R. & RIBEIRO, B. D. A. N. 1999. *Modern Information Retrival*, Addison-Wesley Longman.

BAGLIETTO, P., MARESCA, M., PARODI, A. & ZINGIRIAN, N. 2005. Stepwise deployment methodology of a service oriented architecture for business communities. *Information and Software Technology,* 47**,** 427-436.

BARNICKEL, N., FLUEGGE, M. & SCHMIDT, K.-U. 2006. Interoperability in eGovernment through Cross-Ontology Semantic Web Service Composition. *The Workshop on Semantic Web for eGovernment 2006 Workshop at the 3rd European Semantic Web Conference.* Budva, Serbia & Montenegro.

BAUM, C. & DIMAIO, A. 2000. Gartner's four phases of E-government model.

BEER, D., KUNIS, R. & RÜNGER, G. 2006. A Component Based Software Architecture for E-Government Applications. *First International Conference on Availability, Reliability and Security.* Hawaii.

BERI., B. & VINTAR, M. 2004. Simple Life-Events Ontology in SU(M)O-KIF
Knowledge Management in Electronic Government.

BERNERS-LEE, T. Year. Semantic web - talk at XML 2000 *In:* XML 2000 2000 Washington DC. W3C - http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html

BERNERS-LEE, T., FIELDING, R. & MASINTER, L. 1998. Uniform Resource Identifiers (URI): Generic Syntax. *RFC Editor*

BERNERS-LEE, T., HENDLER, J. & LASSILA, O. 2001. The Semantic Web. *Scientific American 284 (2001)***,** 34-43.

BRUSA, G., CALIUSCO, M. L. & CHIOTTI, O. 2007. Enabling knowledge sharing within e-Government back-office through ontological engineering. *Journal of Theoretical and Applied Electronic Commerce Research,* 2**,** 33-48.

BURK, R. R. 2005. Enabling Citizen-Centered Electronic Government Action Plan. *In:* OFFICE OF E-GOVERNMENT AND INFORMATION TECHNOLOGY, U. (ed.).

BURSTEIN, M., BUSSLER, C., ZAREMBA, M., FININ, T. & PAOLUCCI, M. 2005. A Semantic Webservice Architecture. *In:* HUHNS, M. N. & SINGH, M. P. (eds.) *IEEE INTERNET COMPUTING.* IEEE Computer Society.

CAPGEMINI, IDC, EUROPE, R., SOGETI & DTI 2010. Digitizing Public Services in Europe: Putting ambition into action. *In:* EUROPEAN COMMISSION, D. G. F. I. S. A. M. (ed.). European Commission.

CASTELLANO, M. 2005. An e-Government Cooperative Framework for Government Agencies. *38th Hawaii International Conference on System Sciences.* Hawaii.

CHINNICI, R., MOREAU, J.-J., RYMAN, A. & WEERAWARANA, S. 2007. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language* [Online]. W3C. Available: http://www.w3.org/TR/wsdl20/ [Accessed 02/09/2010 2010].

CHIRCU, A. M. 2008. E-government evaluation: towards a multidimensional framework. *Electronic Government, an International Journal,* 5**,** 345 - 363.

CHIU, D. K. W., CHEUNG, D. H. S. C. & KAFEZA, E. 2007. Towards Ubiquitous Government Services through Adaptations with Context and Views in a Three-Tier Architecture. *40th International Conference on System Sciences.* Waikoloa, HI, USA IEEE CPS.

DIJKMAN, R. & DUMAS, M. 2004. Service-Oriented Design: A Multi-viewpoint Approach. *International Journal of Cooperative Information Systems,* 13**,** 337-368.

DOMINGUE, J., ROMAN, D. & STOLLBERG, M. 2004. *Web Service Modeling Ontology (WSMO) - An Ontology for Semantic Web Services* [Online]. W3C. Available: http://www.w3.org/2005/04/FSWS/Submissions/1/wsmo_position_paper.html [Accessed 02/09/2010 2010].

ERL, T. 2005. *Service-Oriented Architecture: Concepts, Technology, and Design,* NJ USA, Prentice Hall PTR Upper Saddle River.

EVANS, D. & YEN, D. C. 2005. E-government: An analysis for implementation: Framework for understanding cultural and social impact. *In:* OFFICE, A. G. I. M. (ed.).

FANG, Z. 2002. E-Government in Digital Era: Concept, Practice, and Development. *International Journal of The Computer, The Internet and Management,* 10**,** 1-22.

FINNISHGOV. 2008. *The Finnish Centre for Pensions* [Online]. Available: http://www.tyoelake.fi/ [Accessed 15/10/2010].

FORTE, M., DE SOUZA, W. L. & DO PRADO, A. F. 2008. Using ontologies and Web services for content adaptation in Ubiquitous Computing. *Journal of Systems and Software,* 81**,** 368-381.

GHAPANCHI, A., ALBADVI, A. & ZAREI, B. 2008. A framework for e-government planning and implementation. *Electronic Government, an International Journal,* 5.

GIL-GARCIA, J. R. & IGNACIO, M.-M. 2007. Understanding the evolution of e-government: The influence of systems of rules on public sector dynamics. *Government Information Quarterly,* 24**,** 266-290.

GRANT, G. & CHAU, D. 2005. Developing a Generic Framework for E-Government. *Journal of Global Information Management,* 13**,** 1.

GROSOF, B., GRUNINGER, M., KIFER, M., MARTIN, D., MCGUINNESS, D., PARSIA, B., PAYNE, T. & TATE, A., W. VAN DER. 2004. *Semantic Web Services Language Requirements* [Online]. http://www.daml.org/services/swsl/requirements/swsl-requirements.shtml. Available: http://www.daml.org/services/swsl/requirements/swsl-requirements.shtml [Accessed 2007].

GRUBER, T. 1993. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition,* 5(2)**,** 199-220.

GUDGIN, M., HADLEY, M. & ROGERS, T. 2006. *W3C Web Services Activity* [Online]. http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/. Available: http://www.w3.org/TR/2006/REC-ws-addr-core-20060509/ [Accessed 16/11/2010 2010].

GUIDI, C., LUCCHI, R. & MAZZARA, M. 2007. A Formal Framework for Web Services Coordination. *Electronic Notes in Theoretical Computer Science***,** 55-70.

GUIJARRO, L. 2007. Interoperability frameworks and enterprise architectures in e-government initiatives in Europe and the United States. *Government Information Quarterly,* 24**,** 89-101.

GUO, X. & LU, J. 2007a. Intelligent e-government services with personalized recommendation techniques. *International Journal of Intelligent Systems,* 22**,** 401-417.

GUO, X. & LU, J. 2007b. Intelligent E-government Services with Personalized Recommendation Techniques.

HEEKS, R. & BAILUR, S. 2007. Analyzing e-government research: Perspectives, philosophies, theories, methods, and practice. *Government Information Quarterly,* 24**,** 243-265.

HELBELER, J., FISHER, M., BLANCE, R. & PEREZ-LOPEZ, A. 2009. *Semantic Web Programming,* Indianapolis, USA, Wiley Publishing, Inc.

HEPP, M. 2006. Semantic Web and Semantic Web Services. *IEEE INTERNET COMPUTING*.

HUHNS, M. N. & SINGH, M. P. 2005. Service-Oriented Computing: Key Concepts and Principles. *IEEE Internet Computing,* 1.

IMSC. 2007. *ICT-Architecture principles and standards in Australian Government* [Online]. Available: http://www.apsc.gov.au/mac/technology.htm [Accessed 25/02/2008].

INC., U. 2010. *What is unicode? [http://www.unicode.org/standard/WhatIsUnicode.html]* [Online]. Available: http://www.unicode.org/standard/WhatIsUnicode.html [Accessed].

IRANI, Z., AL-SEBIE, M. & ELLIMAN, T. 2006. Transaction Stage of e-Government Systems: Identification of Its Location and Importance. *System Sciences, 2006. HICSS '06. Proceedings of the 39th Annual Hawaii.* Hawaii USA.

IRISHGOV. 2010. *The Irish General Register Office* [Online]. Available: http://www.groireland.ie/ [Accessed 05/07/2010].

JAEGER, P. T. 2002. Constitutional principles and E-Government: an opinion about possible effects of Federalism and the separation of powers on E-Government policies. *Government Information Quarterly***,** 11.

JUNEJA, G., DOURNAEE, B., NATOLI, J. & BIRKEL, S. 2008. *SOA in Government: A Law Enforcement Use Case* [Online]. USA: The SOA Magazine. Available: http://www.soamag.com/I15/0208-1.php [Accessed 15/11/2010].

KOPONEN, T. & VIRTANEN, T. 2004. A Service Discovery: a Service Broker Approach. *37th Annual Hawaii Internation Conference on System Science.* Hawaii USA.

KOTSIOPOULOS, I. & RENTZEPOPOULOS, P. 2009. Bringing Together and Accelerating eGovernment Research in the EU. DG Information Society and Media European Commission.

KRAMER, J. & HAZZAN, O. 2006. The role of abstraction in software engineering. *Proceedings of the 28th international conference on Software engineering.* Shanghai, China: ACM.

LARA, R., LAUSEN, H., ARROYO, S., DE BRUIJN, J. & FENSEL, D. Year. Semantic Web Services: description requirements and current technologies. *In:* Proceedings of the International Workshop on Electronic Commerce, Agents, and Semantic Web Services held in conjunction with the Fifth International Conference on Electronic Commerce (ICEC 2003), 2003 Pittsburgh.

LAYNE, K. & LEE, J. 2001. Developing fully functional e-government: A four stage model. Government Information Quarterly.

LIU, W., HUSNI, H. & PADGHAM, L. 2007. E-Service Composition Tools from a Lifecycle Perspective. *In:* LU, J., RUAN, D. & ZHANG, G. (eds.) *E-Service Intelligent.* Springer.

LU, L., ZHU, G. & CHEN, J. 2004. An Infrastructure for E-government Based on Semantic Web Services. *IEEE International Conference on Services Computing.* IEEE.

LU, S., BERNSTEIN, A. & LEWIS, P. 2006. Automatic workflow verification and generation. *Theoretical Computer Science,* 353**,** 71-92.

LÜ, X. 2007. Distributed Secure Information Sharing Model for E-Government in China. *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing.* IEEE.

LYTRAS, M. D. 2006. The Semantic Electronic Government: Knowledge Management for citizen relationship and new assessment scenarios. *Electronic Government an International Journal,* 3**,** 13.

MADHUSUDAN, T. 2006. A web services framework for distributed model management *Information Systems Frontiers***,** 9–27.

MAEDCHE, A. 2002. *Ontology learning for the Semantic Web,* Bosston, Kluwer Academic.

MCGUINNESS, D. L. 2004. *OWL Web Ontology Language Overview* [Online]. Available: http://www.w3.org/TR/2004/REC-owl-features-20040210/ [Accessed].

MCILRAITH, S., CAO, S. T. & HONGLEI, Z. 2001. Semantic Web Services. *IEEE INTELLIGENT SYSTEMS*.

MEDJAHED, B., REZGUI, A., BOUGUETTAYA, A., OUZZANI, A. M. & TECH, V. 2003a. InfrastructureFor E-governmant webservices. *IEEE Internet Computing*.

MEDJAHED, B., REZGUI, A., BOUGUETTAYA, A. & OUZZANI, M. 2003b. Infrastructure for E-Government Web Services. *IEEE INTERNET COMPUTING*.

MENEKLIS, B., KALIONTZOGLOU, A., DOULIGERIS, C. & POLEMI, D. 2005. Engineering and Technology Aspects of an e-Government Architecture Based on Web Srvices. *Third European Conference on Web Services.* IEEE.

MOMOTKO, M., IZDEBSKI, W., TAMBOURIS, E., TARABANIS, K. & VINTAR, M. 2008. An Architecture of Active Life Event Portals-Generic Workflow Approach *Electronic Government. 6th International Conference, EGOV* Regensburg, Germany.

MUGELLINI, E. P., MARIA. CHIARA. KHALED, OMAR. ABOU. PIRRI, FRANCO. 2005. eGovernment Service Marketplace: Architecture and Implementation. *TED Conference on eGovernment. Bolzano (Italy)*

MUNINDAR, P., SINGH, B. & HUHNS, M. 2006. *Front Matter, in Service-Oriented Computing: Semantics Processes, Agents,* Chichester, UK, John Wiley & Sons, Ltd.

NUNAMAKER, J. F., JR. & CHEN, M. Year. Systems development in information systems research. *In:* System Sciences, 1990., Proceedings of the Twenty-Third Annual Hawaii International Conference on, 2-5 Jan 1990 1990. 631-640 vol.3.

ONTARIOGOV. 2007. *Canada Ontolrio Life events* [Online]. http://www.ontario.ca/en/life_events/index.htm. Available: http://www.ontario.ca/en/life_events/index.htm [Accessed 15/10/2010].

ONTOGOV. 2006. *ONTOGOV project* [Online]. http://www.ontogov.com/. Available: http://www.ontogov.com/ [Accessed 2007].

ORACLE. 2010. *Software Engineering in an SOA Environment* [Online]. http://www.oracle.com/technetwork/topics/entarch/oracle-pg-soa-sw-engineering-r3-0-176714.pdf. Available: http://www.oracle.com/technetwork/topics/entarch/oracle-pg-soa-sw-engineering-r3-0-176714.pdf [Accessed 2/05/2011].

PAPAZOGLOU, M. 2008. Web Services Technologies and Standards. *Computing Surveys*.

PENG, Z., YANZHANG, W. & XUEHUA, W. Year. Research on the Integration in E-government Based on Multi-Agent. *In:* Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, 2006. IEEE.

PERISTERAS, V. & TARABANIS, K. 2006. "Reengineering the public administration modus operandi through the use of reference domain models and Semantic Web Service technologies. *AAAI Spring Symposium, The Semantic Web meets eGovernment (SWEG).* Stanford University, California, USA.

REDING, V. 2006. E-Government Developments, E-Government for All Europeans. *Australia's E-Government Strategy – New Service Agenda.* IOS press.

RETIREMENT-INVESTMENTS. 2007. *National Information Centre on Retirement Investments* [Online]. Available: http://www.nicri.org.au/go/life-events/life-events [Accessed 15/10/2010].

SABOL, T. & MACH, M. 2004. Semantic Web in e-Government. Kosice, Slovak Republic: Faculty of Economics, Technical University of Kosice.

SANATI, F. 2010. *Software Industry Practice Survey* [Online]. Sydney, Australia http://farzadsanati.blogspot.com/p/software-industry-survey.html, University of Technology Sydney. Available: http://farzadsanati.blogspot.com/p/software-industry-survey.html [Accessed 05/11/2010 2010].

SANATI, F. & LU, J. 2008. Semantic Web for E-Government Service Delivery Integration. *IEEE Information Technology Next Generation (ITNG)* Las Vegas USA: IEEE.

SANATI, F. & LU, J. 2009a. Life-event Modelling Framework For E-Government Integration. *Electronic Government: An International Journal,* 1**,** 183-202.

SANATI, F. & LU, J. 2009b. Multilevel Life-event Abstraction Framework for E-government Service Integration. *9th European Conference on e-Government.* London England.

SAUMYA, R. 2002. *How Much Government Snooping Is Okay?* [Online]. http://pcworld.about.com/news/Mar122002id88684.htm. Available: http://pcworld.about.com/news/Mar122002id88684.htm [Accessed 11/11/2010 2010].

SHAHKOOH, K. A. & ABDOLLAHI, A. 2007. A strategy-based model for e-government planning. *International Multi-Conference on Computing in the Global Information Technology.* IEEE.

SHAREEF, M. A., KUMAR, U., KUMAR, V. & DWIVEDI, Y. K. 2009. Identifying critical factors for adoption of e-government. *Electronic Government, an International Journal,* 6**,** 70-96.

SHENG, H. & TRIMI, S. 2008. M-government: technologies, applications and challenges. *Electronic Government, An International Journal,* 5.

SHEU, P. C.-Y. & KITAZAWA, T. 2007. From Semantic objects to Semantic Software Engineering. *Journal of Semantic Computing,* 1**,** 18.

SIEW, P. L. L., P. C. ENG, WAH. LEE. 2006. Web Services Implementation Methodology for SOA Application *IEEE International Conference on Industrial Informatics.*

SKOKAN, M. & BEDNAR, P. Year. Orchestration of public administration services with a use of semantic technologies. *In:* Applied Machine Intelligence and Informatics, 2008. SAMI 2008. 6th International Symposium on, 2008. 209-212.

SMITH, M. K., WELTY, C. & MCGUINNESS, D. L. 2003. *Owl web ontology guide* [Online]. [Accessed 2011].

SOLIMAN, K. S. 2003. E-government: a strategic operations management framework for service delivery. *E-Government.* Bradford, , UK: Emerald Group Publishing Limited.

SOMMERVILLE, I. 2004. *Software Engineering*, Pearson.

SOON, C. A. 2002. Domain Knowledge-Based Automatic Workflow Generation. Berlin: Springer-Verlag.

SPAINGOV. 2008. *Integrated business creation service* [Online]. Available: http://www.cat365.net/inici/default [Accessed 18/10/2010].

STANFORD-UNIVERSITY. 2009. *Protégé* [Online]. Medicine: Stanford Center for Biomedical Informatics Research. Available: http://protege.stanford.edu/ [Accessed 2010].

STOJANOVIC, L., ABECKER, A., APOSTOLOU, D., MENTZAS, G. & STUDER, R. 2006a. The role of semantics in e-government service model verification and evolution. *American Association for Artificial Intelligence Spring Symposia.*

STOJANOVIC, L., APOSTOLOU, D. & STOJANOVIC, N. 2006b. Change management in e-government: OntoGov case study. *Electronic Government an International Journal,* 3**,** 19.

SUNMICROSYSTEMS. 2002. *Java BluePrints Model-View-Controller* [Online]. SunMicrosystems. Available: http://java.sun.com/blueprints/patterns/MVC-detailed.html [Accessed 23/04/2011 2011].

THAYER, R. H. & DORFMAN, M. 1977. *Requirements Engineering,* Los Alamitos, IEEE Computer Society Press.

THE-WORLD-BANK. 2007. *e-government definition* [Online]. Available: http://web.worldbank.org/WBSITE/EXTERNAL/TOPICS/EXTINFORMATIONANDCOMMUNICATIONANDTECHNOLOGIES/EXTEGOVERNMENT/0,,menuPK:702592~pagePK:149018~piPK:149093~theSitePK:702586,00.html [Accessed 4/12/2007 2007].

THOMAS, J. & MCGOUGH, B. 2009. *Kuali Architecture Standards* [Online]. https://wiki.kuali.org/display/KULRICE/Architecture+(Archive+from+Original+KTC+Work. Available:

https://wiki.kuali.org/display/KULRICE/Architecture+(Archive+from+Original+KTC+Work [Accessed 21/04/2011 2011].

TROCHIDIS, I., TAMBOURIS, E. & TARABANIS, K. 2006. Identifying Common Workflow Patterns in Life-Events and Business Episodes. *The second International Conference on e-Government.*

TROCHIDIS, I., TAMBOURIS, E. & TARABANIS, K. Year. An Ontology for Modeling Life-Events. *In:* Services Computing, 2007. SCC 2007. IEEE International Conference on, 2007. 719-720.

TSARKOV, D. & HORROCKS, I. 2006. FaCT++ Description Logic Reasoner: System Description. *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006).* Springer.

TURNER, M., BUDGEN, D. & BRERETON, P. 2003. Turning software into a service. *Computer,* 36**,** 38-44.

UKSALFORDGOV. 2007. *UK Salford Government Life Events* [Online]. http://www.salford.gov.uk/life-events.htm. Available: http://www.salford.gov.uk/life-events.htm [Accessed 15/10/2010].

UMAPATHY, K. & PURAO, S. 2007. A theoretical investigation of the emerging standards for web services. *Information Systems Frontiers***,** 119–134.

W3C. 2004. *OWL-S: Semantic Markup for Web Services* [Online]. http://www.ai.sri.com/daml/services/owl-s/. [Accessed 16/11/2010].

WANG, C., LU, J. & ZHANG, G. 2008. An Ontology Data Matching Method for Web Information Integration. *Proceedings of iiWAS2008.* Linz, Austria.

WOLF, P. & KRCMAR, H. Year. Needs Driven Design for eGovernment Value Webs. *In:* Hawaii International Conference on System Sciences, Proceedings of the 41st Annual, 2008. 220-220.

YU, L. 2007. *Introduction to the Semantic Web and Semantic Web Services,* Boca Raton, FL, USA, Chapman & Hall/CRC.

ZHANG, H., LI, Y.-F. & TAN, H. B. K. 2009. Measuring design complexity of semantic web ontologies. *Journal of Systems and Software,* 83**,** 803-814.

ZHANG, L.-J., JECKLE, M., LARA, R., ROMAN, D., POLLERES, A. & FENSEL, D. 2004. A Conceptual Comparison of WSMO and OWL-S. *Web Services.* Springer Berlin / Heidelberg.

ZHANG, W. & WANG, Y. 2008. Towards building a semantic grid for E-government applications. *WSEAS Transactions on Computer Research,* 3**,** 273-282.

ZHOU, J., ZHANG, S., ZHAO, H. & WANG, M. Year. Towards Semantic Grid-Based Enterprise Information Integration. *In:* Grid and Cooperative Computing GCC 2005, 2005. SGII.

ZIEGLER, P. & DITTRICH, K. R. Year. THREE DECADES OF DATA INTEGRATION—ALL PROBLEMS SOLVED? *In:* 18th IFIP World Computer Congress (WCC 2004), 2004 Toulouse, France.

ZIMMERMANN, O., KROGDAHL, P. & GEE, C. 2004. *Elements of Service-Oriented Analysis and Design* [Online]. IBM. Available: http://www-128.ibm.com/developerworks/webservices/library/ws-soad1/ [Accessed 30/05/2008].