Managing Adaptive Web Services Using Semantic Models and Automated Policy Refinement

By Kevin Carey
Supervisor: Prof. Vincent Wade
School of Computer Science & Statistics
Trinity College Dublin

Declaration

I, the undersigned, declare that this work has not previously been submitted to this or any other University, and that unless otherwise stated, it is entirely my own work.

I agree that Trinity College Library may lend or copy this thesis upon request.

Signed: [redacted]

Decemb

9444

Abstract

There is an increasing demand for web services to be more flexible, in order to suit fast changing business needs and user requirements. Web services need to be adaptive to changes in context, so as to provide web service personalisation, i.e. services that are adaptive to meet user's preferences or that accommodate to changes in the business environment. This thesis investigates a policy-based management approach to achieving adaptive composite services. A key aspect of policy-based management is to be able to specify high-level policies that can be mapped down to low-level policies, in order to manage the adaptive behaviours of web services. This thesis researches how to describe adaptive composite services so as to expose their adaptive behaviours, and how to specify and automatically refine management policies to dynamically manage these adaptive behaviours.

A methodology is introduced for describing and managing adaptive web services. This thesis also investigates the development of innovative tools to support this methodology. A set of integrated tools were developed, which ease the task of describing adaptive web services, and of refining high-level policies in an automated manner to manage adaptive web services. This set of integrated tools undergoes a functional evaluation to validate our approach, and a usability evaluation to assess the usability of these tools.

This thesis proposes a new approach to adaptive web service management, which is accomplished through a novel integration of FSM and Ontology reasoning to automate policy refinement. Thus, this approach enables the auto-generation of refined policies for managing adaptive composed web services. The thesis presents the design of innovative tools that hide the complexity of modelling adaptive web services, and automatically refine high-level policies into auto-generated low-level enforceable policies, for managing adaptive composed web services.

Acknowledgements

Firstly, I would like to thank my supervisor Professor Vincent Wade for all his wisdom, patience, and support. It is through his belief and trust in me that this thesis became a reality. I would also like to take this opportunity to thank my parents for all they have done for me; my brother and sisters Felim, Kathleen and Roisin for their support and words of encouragement.

I would also like to thank Nina for her love and understanding, and with much gratitude to the Burke sisters Michelle and Marie, for helping with my English. And I would like to acknowledge the support of all my close friends, and the friendship of all my friends from KDEG.

Finally, I would like to dedicate this work to Noreen Carey, whose life would have been an endless sea of possibilities.

Table of Contents

Table of Figures

Table of Tables

Abbreviations

AWT – Abstract Window Toolkit

BPEL – Business Process Execution Language

FSM – Finite State Machine

GUI – Graphical User Interface

MAWS – Manage Adaptive Web Service

MVC – Model View Controller

OWL – Web Ontology Language

OWL-S – Semantic Mark-up for Web Services

PBMS – Policy-based Management Systems

RDF – Resource Description Framework

RMI – Remote Method Invocation

SABE – Service Adaptive Behaviour Editor

SMPE – Service Management Policy Editor

SOAP – Simple Object Access Protocol

UML – Unified Modelling Language

URL – Uniform Resource Locator

WSCE – Web Service Composition Engine

WSC – Web Service Composition

WSD – Web Service Description

WSDE – Web Service Description Editor

WSDL – Web Service Description Language

WSDL-S –Web Service Semantics

WSM – Web Service Management

WSMO – Web Service Modelling Ontology

1 Introduction

1.1 Motivation

Increasingly, systems have to cope with rapidly evolving user requirements [10]. To meet these potential requirements, there is a growing need to make web services become more adaptive [11]. This need for adaptive services is increased by the necessity of customising web services to tailor them to meet a client's preference or to restrict web services according to evolving business needs. By developing adaptive applications the same system can satisfy a broader range of requests without requiring manual reprogramming. In other words, the same adaptive web service could be personalised to uniquely suit different users thereby addressing the requirements of a broader range of users without the need to recode.

A second motivation for adaptive web services is the desire to increase reusability [24]. Web services become more reusable when adaptive, since they can adapt at runtime to different circumstances, i.e. they can be used in different contexts, such as customised for different companies and employed for different business models. For example, a service provider could offer a secure service that uses an optimised encryption algorithm for gold subscribers and standard encryption algorithm for bronze subscribers. While another service provider, decides that silver subscribers can afford this optimised encryption algorithm.

One approach to realising adaptive web services is to dynamically compose the services from pre-existing simpler services [12]. In this way, adaptive web services are attained from the adaptive selection of the constituent web services and the workflow across these selected web services [13]. However, this method of realising adaptive web services presupposes that all the elemental services required to perform the required task, and more specifically in the entailed manner[1], are available. Thus, a significant problem here is that the web services may need to be re-composed and alternative elemental services may need to be found.

[1] In general web services are static and can only perform a function in the designed manner; their behaviour is not adaptable to changes

A variation of this approach would be to define web services that are more intelligent, i.e. web services that have multiple behaviours that can be adaptive at runtime. These are services with default behaviours as well as other alternative behaviours which can be used in special circumstances to accomplish a range of service customisation. In order to reason about these adaptive web services, there is a necessity to describe their behaviours formally [1].

However, just modelling the adaptive behaviours is not sufficient; these adaptive behaviours need to be dynamically managed at runtime so as to ensure that web services can dynamically react to contextual changes related to a web application. The key benefit of this dynamic management is that an adaptive behaviour can be realised without necessarily changing the web service composition itself. The problem then becomes how to dynamically choose the correct behaviour for a web service in a particular environment or context, and automate this dynamic management[2] of the chosen adaptive behaviour of a web service.

A number of different approaches to controlling the behaviour of resources or services on the web have been attempted [21][14]. A declarative approach, called policy-based management, allows dynamic control and reasoning about the behaviours of systems or applications [22][8]. Policy-based management systems (PBMS) provide a mechanism to dynamically control the behaviour of systems and services at run time without the need to remodel or recode the managed system. PBMS has been used to manage large complex dynamic systems, such as communication network security and QoS [17] [18]. In recent years, PBMS have focused on managing atomic web services [19], web service security [16], and web service publication and discovery [15]. However, PBMS has not been used to manage the alternative behaviours of composite web services.

PBMS's goal is to provide a high-level means of affecting the overall behaviour of a system, without the need for reconfiguring the system or service manually [42]. Thus, PBMS allow managers to specify high-level policies, in a declarative way, to express a goal or a constraint on a system. However, these policies are abstract and need to be refined to low-level policies. These refined low-level

[2] By management it is meant both the sensing of a particular state of a behaviour and the control of this behaviour

policies affect the local tuning of the relevant components and resources of a system to enforce their goal [50].

A key difficulty in PBMS is the process of refining high-level policies into low-level policies, specifically in an automated manner [51]. Thus, a key challenge when using policies for managing adaptive composite web services is being able to automatically refine these high-level policies into low-level policies to enforce the intended behaviour on the relevant constituent atomic web services.

1.2 Research Goal and Objectives

The goal of this thesis is to propose and evaluate an innovative architectural approach and mechanism, which combines Finite State Machine (FSM) and Ontology reasoning together with policy-based management. This innovative approach supports accurate high-level policy specification and automatic refinement and generation of low-level policies for managing adaptive composite web services[3].

Thus the research objectives of this thesis are to:

- Research the use of Finite State Machine (FSM) and ontological techniques to describe adaptive web services and techniques to support policy refinement.
- Define and develop innovative mechanisms to describe the adaptive behaviours of both atomic and composed web services.
- Define and develop novel mechanisms to specify high-level policies to manage the adaptive behaviours of composed web services and to auto-generate refined (low-level) policies, which can be enforced on the relevant constituent atomic services.
- Evaluate the complexity of designing adaptive service management using this approach, i.e. usability of the approach, and a comparison with other policy refinement approaches.

The thesis proposes to use FSM and Ontology representation techniques to provide the relevant semantics for describing adaptive web services, and proposes to use policies to provide a means to specify the desired adaptive behaviours for a

[3] This thesis focuses on composite services which are made up of sequence of individual adaptive web service invocations

composite service. Furthermore, the thesis proposes to use the semantic description and high-level policy specifications to auto-generate refined (low-level) policies to dynamically manage at runtime the relevant adaptive behaviours belonging to the appropriate constituent atomic services.

1.3 Contribution of Work

This thesis contributes to the state of the art in adaptive web service management. It proposes a novel combination of the use of FSM and Ontology reasoning to automate policy refinement of high-level policies and generate low-level policies to manage adaptive behaviours of composite web services. While each of these technologies has been individually used in some form or another to describe system's behaviours or to manage systems, this approach provides the first novel integration of these techniques to auto-generate refined policies for managing adaptive composed web services.

The thesis presents the design of a set of integrated tools which hide the complexity of both modelling of the FSM and Ontology models, and eases the complexity of authoring policies to manage the adaptive behaviours of composite services. In addition, it automatically refines high-level policies into low-level enforceable policies, based on the tool's ability to automatically generate refined policies.

Publications arising from PhD thesis:

i. **Kevin Carey**, Vincent Wade. "*Using Automated Policy Refinement to Manage Adaptive Composite Services.*" Network Operations and Management Symposium Workshops, 2008. NOMS Workshops IEEE, Salvador Brazil, April 2008.

ii. **K. Carey**, V. Wade. "*Realising Adaptive Web Services through Automated Policy Refinement.*" Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium, Munich Germany, May 2007.

iii. **K. Carey**, D. Lewis, S. Higel, V. Wade. "*Adaptive Composite Service Plans for Ubiquitous Computing.*" Second International Workshop on Management of Ubiquitous Communications and Services, MUCS 2004, Dublin, Ireland, December 2004

iv. J. Keeney, **K. Carey**, D. Lewis, D. O'Sullivan, V. Wade. "*Ontology-based Semantics for Composable of Autonomic Elements.*" Workshop on AI in Autonomic Communications at 19th International Joint Conference on Artificial Intelligence, JCAI'05, Edinburgh, Scotland, July 2005

v. D. Lewis, A. Brady, **K. Carey**, O. Conlan, K. Feeney, S. Higel, T. O'Donnell, D. O'Sullivan, K. Quinn, V. Wade. "*Managed Person-centric Adaptive Services for Smart Spaces.*" eChallenges 2004, eChallenges04, Vienna, Austria, October 2004

vi. D. Lewis, K. Feeney, **K. Carey**, T. Tiropanis, S. Courtenage. "*Semantic-based Policy Engineering for Autonomic Systems.*" Second International Workshop on Management of Ubiquitous Communications and Services, MUCS 2004, Dublin, Ireland, December 2004

1.4 Technical Overview

1.4.1 Technical Approach

This thesis proposes a new approach to describing adaptive web services and managing their adaptive behaviours using policies [1]. In order to accomplish this work it was first necessary to investigate different methods to achieving adaptive web services. In particular, a policy-based management approach was chosen to achieve adaptive web services. It became evident from this investigation that there is a need to formally describe adaptive web services[4] in order to be able to manage them. Hence, the thesis investigates the use of Ontology models to semantically describe the composite services, and Finite State Machine (FSM) as a means to provide a formal representation model for their adaptive behaviours.

The other aspect of the approach taken in this thesis is how to tackle the problem of managing adaptive composite services. A policy-based management approach was the chosen option to manage the adaptive behaviours of these services. However, by using policies some key challenges needed to be tackled:

[4] Typically, semantic descriptions don't provide a formal representation of the service's adaptive behaviours

- the need to ensure that high-level policies specified to manage adaptive services are correctly formed using the correct vocabulary;
- the need for use of low-level policies to enforce the chosen adaptive behaviour;
- the need to automatically refine high-level policies into low-level policies.

Therefore, a survey of techniques for automatically refining high-level policies into low-level policies was performed. From this survey a novel technique was developed, which automatically refines these high-level policies into auto-generated low-level policies using a combination of semantically rich models. This automated policy refinement technique uses the semantic descriptions of composite web services and their adaptive behaviours to generate these enforceable policies.

In order to support the application of these techniques, the thesis proposes a methodology which steps through the key aspects of the proposed approach. The proposed methodology helps users identify the sequencing of activities to perform, and the thesis also defines the tools needed to support these methodology activities.

Rather than developing a single application to provide all of this functionality, a set of integrated tools were developed to suit the different categories of users in the development cycle. This set of integrated tools is divided into:

i. a tool for modelling composite web services and capturing the semantic web description with ontology based descriptions;

ii. a tool to capture adaptive behaviours of web services as FSM; and to aggregate FSM models in order to describe the adaptive behaviours of composed web services;

iii. a tool to define high-level policies for adaptive composite services and to automatically refine these policies generating low-level policies which enforce the correct adaptive behaviours of the relevant constituent web services.

In order to evaluate the tools, a twin evaluation process was executed, which validates the policies generated by the tools and evaluates the usability of the tools. Firstly, the tools were evaluated to validate their ability to describe and manage adaptive web services using two case studies. Secondly, the tool's usability was evaluated; the key issue being investigated here was the ability of these tools to reduce the complexity of developing the relevant models needed for managing the adaptive behaviours of web services.

Finally, a comparison of the novel automated policy refinement approach with other approaches is discussed. This comparison identifies and compares key aspects of automated policy refinement for managing adaptive composite services.

1.4.2 Thesis Overview

Chapter two begins with a state of the art review and an appraisal of current approaches to achieving adaptive web services. Then the various standards and technologies that support the semantic descriptions of adaptive composite web services, are examined. Following this, a review of the prominent architectural designs for achieving policy refinement is presented.

Chapter three illustrates the design and architectural vision of this research and formulates the base requirements for development environments that describe adaptive web services and their managing policies. Based on influences from chapter two, a methodology for describing and composing adaptive composite services and for creating management policies was developed. This provides the foundation for the architectural design of a set of integrated tools that describe adaptive services and their management policies.

Chapter four presents the implementation of a set of integrated tools for describing adaptive composite services, and for specifying high-level policies which are automatically decomposed in order to dynamically manage the adaptive behaviour of composite web services. Based on the design specifications in chapter three, the architecture of these tools and their components is described.

Chapter five presents the results of the case studies and of the usability evaluation carried out on the prototype implementation of the set of integrated tools designed according to the architectural design. Case studies that investigate how well this thesis' innovative approach succeeded in describing adaptive web services and their management policies are also presented. Three separate usability evaluations were conducted with different groups of computer science graduates, and each time results were recorded using standard usability engineering questionnaires. Also as part of the evaluation, a comparison was performed between this approach and other approaches to automated policy refinement, previously identified as state of the art.

Finally, the thesis concludes with a description of the objectives and achievements of this research, a summary of key contributions attributed by this research and its limitations, as well as a discussion of the pertinent future work to continue and broaden this research.

2 State of the Art

2.1 Introduction

Adaptive web services have gained a lot of interest in recent years due to the need to continually adapt web services to rapid changes in the market where there is a necessity for web services to be customised to meet business needs [20]. Furthermore, there is a need for web services to be adaptable to meet user's preferences, i.e. web services that can be personalised to meet user's needs. There are different ways to achieve adaptive web services.

Introduction to adaptation is first described, detailing the different methods of adaptation and adaptive systems. An investigation into different approaches to achieving adaptive web services is presented, together with a discussion of how the different approaches compare in their achievement of adaptive web services. A survey and a discussion of the different modelling languages that can be used to formally describe adaptive web services are also presented. This is followed by a survey of different approaches to automated policy refinement. Finally, choices of technologies for representing obligation policies that can be used in the proposed approach are presented.

2.2 Adaptation

Adaptation is the process in which an adaptive system adapts its behaviour to each request based on inputs and changes in context [81]. Many software systems need to be customised before using them, either to satisfy the user's needs or to provide interoperability with other software systems. This customization is a form of adaptation especially if done in an automated manner. There are many reasons for adaptive systems, and such needs are growing with the increasing development of distributed systems and its ad-hoc nature [82]:

- Reuse of components and the significance of component integration, instead of programming from foundation;
- New paradigms for distributed computing based on mobile technologies;
- Creation of context-aware smart environments - ubiquitous environments;

- Increase development of dynamic user centred web sites;
- Growing use of real-time interactions and multimedia based content in groupware and collaboration systems.

Therefore, many forms of adaptation in software systems have emerged in diverse areas, from context aware system to web personalisation. For example, systems that register their locations or physical devices and change their behaviour accordingly are deemed context aware system [86]. Systems that would change their web presentation to different users depending on their preference or usage history are called web personalisation [83]. Adaptive e-learning systems can provide different sets of modules for a course, depending on user's knowledge and preference [84]. Adaptive middleware are frameworks that would change its configuration to better suit its running applications, for instance to provide a better QoS [85].

From the different methods of adaptation, a few distinguished methods have been noticed, such as adaptation through composition, reflection, and policy-based management. Adaptation through composition composes its components on the fly to produce a system that would achieve the desired task. Reflection allows for other systems to inspect and modify the behaviours by calling unpublished functions of a running system. Policy-based management provides a means to automatically change the behaviour of a running system when a particular policy rule is triggered.

The main advantage of adaptive systems is that they can accommodate a broad range of users. Adaptive systems can be personalised to suit the user's needs; whether these personalisation is driven by the user (being able to select a specific course module), or driven by the system (being recommended a particular book that user might wish to buy). Another advantage is that adaptive systems are customizable to integrate or interoperate with other software systems. For example, composable middleware frameworks can ensure safety interactions for ubiquitous applications [87].

One of the concerns with adaptive systems is security. By providing the flexibility to change the system, it can also cause unforeseen security issues to surface. Customisation has the disadvantage of empowering the users with the ability to miss-configure the system. Another disadvantage of adaptive systems is that if not designed properly can lead to conflicts between users trying to customize the same shared adaptive system.

2.3 Achieving Adaptive Web Services

As mentioned previously, there is an increasing demand for web services to be more intelligent, i.e. to be adaptive to different situations. This thesis defines adaptive web services as web services that can dynamically change their internal behaviour according to changes in context relating to them. By changing their internal behaviour, adaptive web services do not change their overall functionality. Instead, adaptive web services allow selectively overriding or customising core functionality of web services when needed, which changes how web services achieve their tasks. Three possible different candidate approaches to achieving adaptive web services were identified:

i. In-house Development, where web services are built with embedded logic;

ii. Dynamic Service Composition, where service composer engines are used to re-compose the necessary web service for a particular change in context;

iii. Policy-based Management, where policies are used to choose the available alternative behaviours of web services.

2.3.1 In-house Development for Achieving Adaptive Web Services

Adaptive web services can be achieved by embedding intelligence logic within web services. Web services can be created with built-in logic that allows them to be adaptive to their environment through different types of inputs. In other words, intelligence logic can be added to web services to change their behaviour based on the values of the different inputs of a web service.

For example, developers can create a notification service that contacts recipients with a designated message. Suppose such a service had inputs: message, email address, and phone number. Let's say that this service is then enhanced by adding intelligence logic so that it will only contact a recipient by phone if a phone number is provided, and it will only contact a recipient by email if an email address is provided as an input. Where both email address and phone number are provided, then, the service will try to contact by phone first, and if the recipient is not reached, it will send an email contact message.

The In-house approach allows web services to be adaptive by hardwiring intelligent decision logic within web services, which changes their behaviours based on input values. The benefit of this approach is that it allows developers to quickly add intelligence to web services, which reacts to different input values. Developers do not have to learn about nor follow any framework for adaptive web services. They are free to create adaptive web services whichever way they want. Their intelligence code is hidden within the web service, which prevents clients and competitors from prying into the secrets of the adaptive logic.

However, the In-house approach has several drawbacks, such as the lack of a framework for adding intelligence logic to web services in a standard manner. Without a framework for exposing the descriptions of the web service's intelligence logic, it becomes difficult to formalise or generalise this approach. This approach does not provide a formal means for clients to be aware of the service's internal intelligence logic.

But more importantly, by hard-wiring the intelligence within the service, it makes it difficult to update or change it to accommodate different or new unforeseen scenarios. For example, what if a user wanted to use the notification service to notify his recipients by phone during business hours, or else contact them by email? This service would have to be re-implemented and redeployed. The In-house approach only allows for web services to be adaptable to users but not to service providers since the intelligence logic is tied to the service's inputs and are only executed when a service is invoked.

2.3.2 Achieving Adaptive Web Services by Dynamic Service Composition

Another approach is to have a collection of elemental web services which can be grouped together to perform a more complex task. These elemental services are atomic web services that perform specific tasks. These web services are then composed together, during runtime, into a composite service tailored to perform a particular complex task.

A web service composition engine (WSCE) is used to automate this composition process. There are many frameworks available to achieve dynamic service composition such as: Artificial Intelligence (AI) planners [24][4], eFlow [25], Petri-Net [26]. AI planners will be used as the default approach for illustrating the dynamic service composition approach to achieving adaptive web services.

An AI planner composes a new web service to satisfy the requested service that accomplishes a particular task or meets specific criteria [24]. AI planners compose web services from a pool of web services available to it. Being given the desired set of outputs and available inputs, AI planners use backward chain rules to find the appropriate set of services that will satisfy these conditions. More sophisticated AI planners would take other factors into consideration, such as QoS or context information, during the web service composition process.

Before proceeding, let us consider the following web service example: a report service, which generates a report document and can be used with an email service to send this document to all the clients via email. These services are combined together in this manner to allow a courier company send its delivery report to the sender or recipient.

It is envisaged that adaptive web services could be achieved through dynamic re-composition of web services. For example, WSCE can be used to recompose web services in order to adapt them to meet some new criteria or to be used in different contexts. Suppose a service was needed to send its delivery report of precious cargo in a secure manner. In order to adapt to the new context, WSCE can recompose the report service mentioned above with security criteria specified. This would result in a new report service, namely a report service that includes an encryption of the document before sending it out to the clients via email.

This dynamic service composition approach to realising adaptability has the advantage of allowing developers to create simple web services for specific tasks. Then these web services can be dynamically combined at runtime to perform a more complex required task. In general, these web services (used in these compositions) are not adaptive, but rather the adaptability is achieved through dynamic re-composition. The benefit of this approach is that atomic web services can be implemented in a straightforward manner with minimal accounting for changes in context. Therefore, developers do not have to worry about such complexities and would leave it to the dynamic service composers to handle the task of enabling adaptive web services through re-composition. This approach provides a framework for adaptive web services, and the intelligence logic is in the WSCE which is external to the services used. Thus, services do not need to be recoded.

The difficulty with the dynamic service composition approach is that the composition process is time-consuming, and web services need to be recomposed every time, in order to adapt to different scenarios. Dynamic service composition approaches, such as AI planners, need to be aware of all of the different web services available to them before performing service composition. To achieve a successful adaptive web service, a WSCE needs to have variants of different web services for the composition process. Variant services are services that perform the same functionality but with an alternative behaviour. For example, if a data storage service stores data in a relational database, then a variant service could be one that stores data in an xml database, and another variant service could store data in a flat file.

The downside to this approach to adaptive web services is that it is still limited by the number of different service variants available at runtime. The necessity to have service variants places an extra burden on the developers to maintain an increased number of services. Furthermore, the addition of service variants causes the decision tree to grow exponentially large, and can add extra complexity to the composition logic. This can cause performance degradation when composing web services, due to the large decision tree and the added complexity in the composition logic when searching for the correct service among all web services and their variants.

To illustrate this point, take the example of Windows users preferring their report in Microsoft Word, Linux users in PDF, and Mac users reading their report in HTML using a web browser. Then the WSCE needs to have three different variants of the report service available, so that it can either produce a report document as Word document, PDF, or HTML; otherwise, it might not accommodate all these users. Thus, it should be more appropriate to use this approach to achieve a particular composition and fine-tune the services using another method rather than recomposing web services over and over again in order to satisfy all the criteria of the desired composite web service.

This approach could also use a script to add intelligence to the service's composition, so that composite services are adaptive to change in context without re-composition. In other words, to add conditional logic in the workflow of composite web services, which can enable web services to react to change in context. These workflow scripts could allow composite web services to be adaptive by instructing the composite service to self recompose, i.e. to change the service selection of its constituent services, when a predicted condition arises, thus allowing the composite service to be adaptive.

The use of this workflow script has the benefit of enabling composite web services to be adaptive without re-composition. Thus, adaptive composite services will not have to rely on a centralised model; adaptive composite service could be deployed in a distributed architecture using this approach. Developers still does not have to worry about creating intelligent web services.

The downside is that this approach is still limited by the number of different service variants available at the time. Furthermore, the added complexity in the service's workflow can introduce unforeseen problems, as well as the need to manually change the script to handle new contexts.

2.3.3 Achieving Adaptive Web Service using Policy-based Management

A different approach to adaptive web services is to define web services that contain all the variants within them, i.e. web services that have multiple alternative behaviours, termed adaptive behaviours, and which can be managed dynamically. However, these adaptive behaviours are managed externally during runtime in order to change how the services accomplish their tasks.

Policy-based management systems (PBMS) can be used for controlling the state of a system and for controlling the devices of a network, using policies [22]. These policies manage the configuration and behaviour of one or more entities within a managed system to achieve some overall behaviour. PBMS are concerned with the overall behaviour of the managed system and adjust the policies that are in effect based on how well the system is achieving its policy goals. These policies are used to control the behaviour of the managed system in a predictable and consistent fashion.

In the case of adaptive web services, it is envisaged that policies could be used to control their internal adaptive behaviours.

PBMS provide great flexibility in their approach to managing a system, where they can change the system's behaviour without the need to remodel or recode the managed system. This means that the intelligence logic can be expressed externally through policies. Thus, web services can be manipulated dynamically according to their policies without the need to recode or re-implement them. By specifying the intelligence logic of web services as policies, PBMS can dynamically modify the behaviours of these adaptive web services at runtime, i.e. allowing adaptive web services to dynamically adapt to changes in context.

However, in order to reason about these adaptive web services, there is a necessity to describe these web services and their adaptive behaviours in a formal manner. By having a formal description of adaptive web services, PBMS can reason about them and policies can be correctly specified using the appropriate vocabulary.

A downside to PBMS is that the use of policies can be complex and cumbersome, such as when authoring several policies for a large composite web service, and when maintaining them. Furthermore a service manager needs to be a policy expert in order to use this approach. Ideally this approach needs to have a framework that is easy to use, so that neither service developers nor service managers need to be policy experts.

Furthermore, managers specifying policies should be able to do so at a high-level of abstraction, without needing to know the intricacy of the adaptive behaviours belonging to web services. Therefore, it is vital that policies can be specified at high-level using abstract elements, and that policies used to enforce these behaviours should be generated or mapped automatically. This is a major issue among PBMS and this issue becomes obvious when asked how to map low-level policies assigned to elemental services, to high-level policies specified for the parent composite service. Or more precisely how to automatically refine high-level policies, assigned to a composite service, into enforceable policies for managing the adaptive behaviours of the constituent adaptive web services.

2.3.4 Discussion of Approaches to Achieving Adaptive Web Service

When comparing the three approaches to achieving adaptive web service, it is clear that the first approach – The In-house approach – is the weakest one. While the In-house approach is beneficial when performing quick web service prototypes, it is hard to maintain them, and this approach does not scale well when many adaptive behaviours are being created. The cause for these limitations is that the intelligence logic of the adaptive web services provided by this approach is hardwired and hidden within the service. Such a method for adaptability prevents the intelligence logic being modified without recoding the web service itself.

The dynamic service composition approach provides a framework for adaptive services through the use of service variants and dynamic re-composition. This approach offers a framework that provides developers with a means to better manage complex adaptive web services and maintain them in the long run. One developer can be responsible for composing web services while other developers can dedicate their time to creating the elemental services used in the composition. This approach separates the burden of adaptability of web services from the development of web services because adaptive web service is achieved through re-composition.

But the need for recomposing web services every time there is a change in context can be time-consuming. In addition, the necessity for several different web service variants to be available during the composition process, can affect scalability, add complexity to the composition decision tree, and cause performance degradation.

Ideally, it would be more productive for the elemental web services, used in the composition, to be built with intelligence. These intelligent web services would be able to fine-tune their behaviour to suit different situations. Then at runtime these composite web services can accommodate to certain changes in context without the need to recompose. The outcome is that less strain will be placed on the WSCE since multiple re-composition of web service will not be necessary.

The policy-based management approach is based on this ideal; where adaptive web services are composed together[5] to perform a specific task, and policies are then

[5] Composed together by some other means

used to manage the adaptive behaviours of these web services to further fine-tune them to suit the user's preference. Web services become adaptive by adding alternative behaviours within them much like the in-house approach. However, these adaptive behaviours are not hard-coded intelligence logic but, instead they are exposed so that they can be managed though the use of policies. By using a policy-based management approach, web services can be dynamically adaptive at runtime, and since the intelligence logic is external, it can be updated without the need to re-code these web services.

The policy-based management approach removes the burden of fine-tuning from the composition engine which would improve its performance and scalability. But this approach adds responsibility to developers of elemental web services to ensure that extra adaptive behaviours are included in these services. One of the benefits of the policy-based management approach is that it provides a separate management interface for controlling these adaptive web services, which is not tied to their dataflow. However, in order for the policy-based management approach to work successfully with adaptive composite web services, policies need to be specified to the composite service as high-level policies, and they need to be decomposed into low-level policies to the relevant constituent services.

A common problem with web services is upgrading them while maintaining backward compatibility. Consider a case where a composite web service has some of its elemental services upgraded but there is a need to keep their service backward compatible. In such a scenario, the dynamic service composition approach would accomplish this task by having two separate service variants that can be chosen during composition, but this could cause confusion. Whereas the policy-based management approach overcomes this obstacle by adding the new feature of the elemental services as adaptive behaviours that can be managed using policies.

When considering the policy-based management approach to adaptive web services, two questions, which need further investigation, were recognised. The first question is how can adaptive web services be defined or described so that they are better understood, i.e. reasoned, or managed by policies. The second question is how these adaptive web services can be controlled during run-time according to their specified high-level policies (goals). This is vital in the case of adaptive composite services. Policy refinement technique claims to provide answers to this issue. A survey is provided in section 2.5.

2.4 Defining Adaptive Web Services

Adaptive web services are web services that can dynamically change their behaviour due to a change in context as they perform their tasks. One of the approaches mentioned above, the policy-based management approach, envisages that adaptive web services have internal adaptive behaviours, which are not hardwired to some internal intelligence logic, but instead exposed so that they can be controlled externally. By exposing the adaptive behaviours through a management interface, they can be controlled externally by a PBMS.

However, one of the shortcomings with the policy-based management approach (discussed in section 2.3.3) is the necessity of a formal and semantically rich description of the adaptive web services. In order to select a suitable modelling language or combination of modelling languages, a brief investigation comparing different modelling languages for describing adaptive web services was conducted. This investigation is presented in this section.

Web services can be used in a loosely coupled manner or composed together to perform a more complex process [31]. So, it is natural to expect that these adaptive web services can be either atomic or composite web services. Adaptive composite web services would encapsulate the combination of composition and policy-based management approaches to adaptive web services.

In order to reason about adaptive web services, they need to be defined with a rich semantic description. This description needs to encapsulate the key aspects of adaptive web services. It could be suggested that four aspects important for defining adaptive web services are:

i. The information necessary for users or client applications to access a web service, i.e. the functional aspect of web services;

ii. The semantic aspects of a web service to handle the automatic discovery and interoperation of web services is through a semantic rich description language;

iii. The composition description of composite web services, thus allowing management system to reason about the composition and the constituent web services of composite web services;

iv. The description of the adaptive behaviours within web services, i.e. a description of the management interface for reasoning about these adaptive behaviours.

There are several existing technologies for describing web services both syntactically as well as semantically. A survey of different technologies - modelling languages - for defining adaptive web services is presented below. Six different modelling language candidates were identified from the three technology domains: WSDL, Ontology, and UML. Each modelling language candidate is presented below, while measured against the criteria identified above.

2.4.1 Describing Adaptive Web Services with WSDL Technology

This section examines WSDL and BPEL as two potential candidates for describing adaptive web services from the WSDL technology domain.

Web services can be defined as self-contained, self-describing, modular applications that can be published, located, and invoked across the web. But in order to publish them on the Internet, one necessary step is to describe web services in a standard manner, which allows clients to discover and interact with them uniformly. Web Services Description Language (WSDL) is *de facto* industry standard for describing a web service's interface [32], and interaction with these web services is achieved using Simple Object Access Protocol (SOAP) messages [33] as described in the web service's associated WSDL description. WSDL has been standardised by W3C [31] and it has even been adopted by other technologies as their grounding, e.g. OWL-S and BPEL among others.

WSDL is an XML-based language created for describing web services. It describes the syntactic information of a web service in a machine-readable document format (XML) [27], and provides a platform-independent model for describing web services, which defines their public interfaces and how to invoke them. A WSDL document defines the public available functions (port types), message formats and protocol bindings that are required to interact with web services. Port types are abstract collections of supported operations that can be performed. Message formats define how to interpret the data types passed in messages. Protocol bindings define how to map messages onto concrete network transports.

WSDL is a strong candidate for the first criterion – describing the functional aspect of web services. However, it does not meet the second criterion due to the lack of semantics in its description; WSDL lacks domain specific data definitions, operation restriction definitions, operation sequence definitions, data mediation definitions and behaviour mediation definitions. Furthermore WSDL was not designed to describe the interactions or workflow of web services, i.e. their composition. WSDL describes web services as a black box and lacks semantics in its modelling language to describe any of the web service's internal behaviour, much less their adaptive behaviours. Although WSDL did not satisfy most of the criteria, it will not be discarded since WSDL is used as the foundation modelling language and combined with other modelling languages to enrich the description of web services.

WSDL-S [80] is a lightweight approach for adding semantics to web service descriptions, specifically described in WSDL. It provides simple extensions to WSDL thereby allowing semantic descriptions of actions, inputs, outputs, preconditions and post-conditions of a WSDL operation.

While web services described by WSDL-S can be orchestrated with hard coded applications, WSDL-S lacks semantics in its description to describe the workflow information of a web service composition. Therefore, WSDL-S does not meet the third criterion since it is not suited to describe the composition of composite web services. Although, the objective is not to describe all of the service's behaviour, it is still required that the internal adaptive behaviours of web services be described so that they can be reasoned with. WSDL-S adds extra semantics but it still does not contain semantic definition to describe web service's internal behaviours.

Business Process Execution Language (BPEL) describes the interactions and workflow of web services within a composite web service [34]. BEPL is built on the WSDL 1.1 specification. It offers a rich process description notation to describe service process behaviours and data dependent behaviours, as well as, exception conditions and orchestration for peer-to-peer interaction between constituent services grounded in WSDL. In BPEL web services can be modelled in two ways, as either executable or abstract (partially specified).

BPEL is an orchestration language, which specifies composite services as services that contain message exchanges with other services, such that the message exchange sequences are controlled by the orchestration engine. BPEL defines an interoperable integration model that should facilitate the expansion of automated process integration. BPEL model includes constructs to describe the control flow and dataflow of service's composition. Control flow description for composite services is defined using control constructs, including *sequence*, *choice* and *if-then-else* among many others.

BPEL is a powerful language that has a semantic description rich enough to describe web services semantically, thus satisfying the second criterion. And since BPEL is grounded in WSDL, it also satisfies the first criterion though the use of WSDL. The semantic workflow description used in BPEL to describe a web service's composition allows BPEL to meet the third criterion.

Again, the objective here is only to semantically describe the internal adaptive behaviours of web services in order to manage those adaptive behaviours using policies. It is possible to attempt to extend BPEL to accommodate the description of a management interface for controlling these adaptive behaviours. However, the semantic definition for this management interface still needs to be formalised.

2.4.2 Describing Adaptive Web Services with Ontology

Ontology is a declarative representation of concepts within a domain, and the relationships between those concepts [89]. Ontology can be used to semantically describe the entities within a domain and to reason about them. It provides a shared vocabulary which is structured and computer readable that can be used to model a domain. Furthermore, ontology models are extendable so that they can continue to model their domains as these domains changes.

Ontology was initially used in Artificial Intelligence domain to create computational models for enabling automated reasoning. Since then, the concept of ontology has expanded into other domains and it is widely used in Semantic Web, Biomedical Informatics, Data Federation, and System Engineering.

Ontology describes the type of objects, their properties, and relations. Ontology focus on entities described as classes, i.e. classes are used in ontology to describe the concepts in a domain. Ontology allows for the modelling of the relationships between concepts. For example, the relationship of two concepts which are parent and child of each other can be expressed as follows: the parent class is a *superclass* of the child class, and the child class is a *subclass* of the parent class. Each class can also have attributes that will provide them with properties or restrictions. Another benefit of ontology is that it can model instances of concept classes and an instance of an ontology class can have its property set with particular values.

An ontology model is a unique list of concepts for a particular domain expressed in an ontology representation language. The ontology language is an important element in building the ontology model. The ontology language must have a grammar with formal constraints which is used to control how the ontology terms are related to each other. For example, an ontology language would have grammar definitions to allow two concepts which are parent and child to be expressed as subclass or superclass of each other.

There are many ontology languages emerging which were created either for general purpose or for modelling concepts of a particular domain. Open Biomedical Ontologies (OBO) [97] is a collaborative effort in creating an ontology language to describe science-based ontologies, which is shared across different biological and medical domains. Knowledge Machine (KM) [98] is a frame-based language with first-order logic semantics, which is used for encoding knowledge representation bases. One of the main markup ontology languages is the Web Ontology Language (OWL) [35]. OWL is an ontology language that has been recommended by W3C for authoring, publishing, and sharing ontologies on the World Wide Web. This ontology language is derived from the DAML+OIL Web Ontology Language and is based on RDF/XML markup scheme.

The advantages of Ontology are that ontology is reusable and flexible. Ontology models can be published and share between communities and even domains. These ontology models are flexible to be expanded as needed, for example if shortcomings are spotted. Since OWL is based on XML, ontology models described in OWL are human readable and it is possible to use off the shelf tools for processing and querying them such as XPath and XSLT. There are also many tools for OWL, such as Protege [95], Jena [46], and Pellet [96].

The disadvantage of ontology is that it has a high reasoning complexity. When creating an ontology model, a choice needs to be made between expressiveness and scalability, i.e. between a light model for inference performance or a full model for model completeness. Ontology models described in OWL can be very verbose due to its XML nature, which can be difficult to read, creating large size files, and making parsing slow.

Subsequently, a couple of potential ontology-based candidates for describing adaptive web services come from the semantic web service domain, which provides a semantic orientation for the description of web services. It uses an ontological language to add semantics to web service descriptions. By using richer semantic descriptions to describe web services, it is possible to better advertise and subsequently discover web services and supply a better solution for the selection, composition and interoperation of web services [36]. The main approaches investigated for describing semantic web services using Ontology are OWL-S [36] and WSMO [29].

Web Service Modelling Ontology (WSMO) is an ontology for semantically describing (various aspects related to) semantic web services. WSMO describes semantic web services from the client as well as the service provider point of view [29]. WSMO uses WSML [30], a semantic web language targeted specifically at semantic web services. WSMO core elements are: an *Ontology* element that provides the concepts and relationships used by other elements; a *Goals* element that defines the user's objectives; *Web Services* descriptions that define various aspects of a web service; *Mediators* which bypass interoperability problems.

OWL-S (formally DAML-S) [36] is an Ontology model for describing web services expressed in Web Ontology Language (OWL) [35]. It contains a set of rich class representations and rich typing that allows semantic web services to be understood by machine processible systems. This semantic description complements current web service description, for instance, WSDL. The OWL-S ontology semantically describes web services in three parts: *service profile*, informs the service capabilities; *process model*, describes the service process; and *grounding*, provides a mapping for web services and their parameters to their WSDL descriptions.

OWL-S and WSMO are the two major efforts that share the same vision that ontology is essential to support automatic discovery, composition and interoperation of web services. WSMO model is more complex and has more semantics for the area of interoperability through its semantic definitions such as mediators. But OWL-S is more mature in the area of semantically describing composite web services (choreography) and grounding, and easier to understand due to its single view of the web service [7].

When examining the OWL-S model, it can be noted that the *process model* ontology represents services as processes. Each process (service) has its parameters (IOPE: inputs, outputs, preconditions, and effects) semantically described using Ontology definitions, and grounded (mapped) to a WSDL description. Hence the semantic aspects of web services are described by the OWL-S process model, and the syntactical aspects of web services are delegated to their WSDL description. Furthermore, the OWL-S process model categorises a process into three process types: atomic, simple and composite processes. An atomic process is a directly invocable process that executes in a single step from the perspective of the service requester. A simple process is not invocable and is not associated with service grounding; it provides a means of abstraction for the other types of process.

A composite process is constructed from other processes that can be either an atomic or a composite process. The composition of composite services is described as control flow and dataflow. Control flow of a composite process is described using control constructs, which among others are *sequence*, *choice* and *if-then-else*. The data flow of a composite process is described by providing ontology definitions for mapping the inputs and outputs, as well as the preconditions and effects of the relative processes.

In the same manner as BPEL, OWL-S was designed with the intent to describe elemental services as a black box. Thus, OWL-S is not designed for describing internal adaptive behaviours. However, OWL-S uses a rich semantic description model based on Ontology to describe the different aspects of web services and since OWL-S is an Ontology-based language, it is possible to extend the model to describe the internal alternative behaviours of these adaptive web services. For example, these alternative behaviours could be described as a sequence of actions. But such Ontology definition for describing adaptive behaviours needs to be formalised and extensively adopted in order for developers to use it.

2.4.3 UML Technologies for Describing Adaptive Web Services

So far the modelling language candidates surveyed were able to describe web services and their composition, but were not designed to describe their adaptive behaviours. Unified Modelling Language (UML) is a standardized modelling language used in the area of software development, with the objective of describing systems and their behaviours [37].

UML has a set of graphic notation techniques which is used to create visual models of software applications. Using UML diagrams developers can model and visualise the static and dynamic views of a software system. Specifically, UML diagrams can represent the structure, behaviour, and relationship of object-oriented software applications. UML is extensible through UML profile, which can be used to extend the model by defining additional diagram types or notations.

There are several commercial and non-commercial UML tools. Tools such as PowerDesigner [91] and Rational Software Architect [92] are just a few of the commercial tools available. UML tools such as ArgoUML [93] and Umbrello UML Modeller [94] provide the same functionality but they are open source and can be used with a non-commercial license. These tools can also provide features that would assist developers, features such as code generation for the UML models and reverse engineering by deriving UML models from source code. Most of these UML tools allow UML models to be exchanged between them by using XMI (XML Metadata Interchange) interchange format.

UML has different types of notation techniques to create abstract models of specific systems; these notations are divided into three categories: structural, behavioural, and interactional. Behavioural diagrams emphasize what must happen in the system being modelled, i.e. system behaviour. The following three diagram types are behavioural diagrams:

- Finite State Machine: standardized notation to describe system behaviours, from computer applications to business processes [71].

- Activity Diagram: represents the business and operational workflows of components within a system. An activity diagram shows the overall flow of control [72].
- Use Case Diagram: shows the functionality provided by a system in terms of actors, their goals represented as use cases, and any dependencies among those use cases [73].

Use Case Diagram does not seem appropriate to describe web services and their adaptive behaviours. This section examines Finite State Machine and Activity Diagram as two potential candidates for describing adaptive web services from the UML behavioural diagrams.

Finite State Machine (FSM) models are a set of semantic concepts that can be used to model discrete behaviours of any software system [71]. They can be used to specify the behaviour of individual entities or to define the interactions between entities. A FSM model shows possible *states* that a system or a component can have, and which *events* can cause the state to change. Events can represent timers, counters, aspects of service invocation (e.g. service invoked, response sent), or changes in the context in which a system is operating. A change of state is called a *transition* and all of these modelling components are encapsulated by (grouped together under) a *finite state machine*.

In the current UML specification a clear distinction between state machine semantics and its graphical notation is established [37]. Two major motives can be seen for using FSM are: (i) for visualising the behaviour of a system, (ii) as information model so systems can reason about it. With regards to visualisation, FSM have been standardised and commercialised into applications, like Poseidon [79], which are used by developers to model object-oriented applications. Poseidon is one of many UML tools that have a graphical user interface which allows for the visualisation of a system's behaviour modelled as FSM.

When reasoning about a system, FSM has been used in diverse areas such as:

- Embedded systems, where FSM is used for modelling the behaviour of microcontrollers;

- Computer games, where FSM is used as the artificial intelligence control technique, say for controlling the actions of characters in the game;
- Policy management systems, where FSM is used as information model to reason about the system.

The advantages of a FSM are its capacity for expressing descriptive behaviours; both the entities and the interactions between the entities. It can model deterministic behaviours; where for each state there is exactly one transition for each possible input, and non-deterministic behaviours; for each pair of state and input symbol there may be several possible states. Furthermore, it is the simplicity of the FSM that facilitates developers in quickly adopting it. FSM are quick to design and relative flexible to model behaviours. Also, the predictability (in deterministic models) of FSM allows for easy understanding and testing of the FSM models.

The main disadvantage of a FSM is that the number of states can increase very quickly when modelling a complex behaviour. Behaviours with a large number of inputs can lead to a state explosion problem, as each required state or state path must be repeated for all possible input values. Larger FSM can be difficult to manage and maintain without a well thought out design. FSM can only be used to describe systems' behaviours that can be decomposed into separate states with well defined conditions for state transitions. This means that all states, transitions and conditions need to be known up front and be well defined. Furthermore, even though it is rich in semantics, it does not allow the use of Ontology. Thus, it limits the FSM model from can expanding its semantic definitions using UML profile to accommodate customisation of the FSM model but it is not as easy and commonly achieved as with Ontology.

When examining the use of FSM to describe adaptive web services, it can be seen that it is not suited to describe the syntactical aspect of a web service (first criterion) due to its lack of web service terminology. Also, when describing the semantic aspect of web services (second criterion), FSM was not designed to describe web services syntactically, nor semantically. However, FSM was designed to describe system behaviour and so it could be used with the rest of the UML definitions to describe web services. But this is not ideal since more popular languages in this domain such as OWL-S and BPEL already exist. Thus, FSM is not considered to be a good candidate for describing semantic properties of web services.

FSM could be used to describe the composition of composite web service (third criterion) where services can be represented as *states*, and control flow constructors can be represented with *transitions*. But describing web service composition using FSM can be tedious. Other modelling languages have more refined semantic definitions for describing control flow and dataflow of composite web services. On the other hand, the adaptive behaviours of web services (fourth criterion) could be described formally as FSM, where each of the adaptive behaviours within a web service could be formally represented by a FSM. These adaptive behaviours are internal to web services, and its structure doesn't conform to any standard. Therefore, the use of states and transitions, from the FSM model, is ideal since it can describe most component behaviours.

In UML, Activity (Graph) Diagram models the sequence of actions between different components. Activity Diagram is an extended view of the FSM, i.e. it is a special case of a state machine that is used to model processes focusing on the sequence of actions taken and their conditions. It uses all of the modelling constructs from FSM and it also has a few semantics of its own. Since Activity Diagram is based on FSM, it has the same benefits and drawbacks found with FSM.

2.5 Representing Management Policies

An *obligation* policy defines a set of actions that must be performed on a target, when triggered if the conditions are satisfied [21]. Obligation policies are normally used for managing the resources or modifying the behaviours of a system dynamically. This can be done to either fine-tune the system's overall performance, e.g. to satisfy QoS requirements [9], or to configure a system to achieve a particular goal, such as to satisfy new criteria. *Obligation* type policy is a suitable policy type for managing adaptive services, i.e. it is suggested that obligation policy is a suitable policy type to represent the management policies [74].

However, if a policy-based management approach is to be used to manage adaptive web services, then a suitable policy language needs to be specified to manage the adaptive behaviours of these web services at runtime. When specifying policies to manage adaptive web services, they should be high-level policies that would have abstract values, while during runtime, the policies used by the PBMS to manage adaptive services needs to be enforceable policies, i.e. low-level policies with their aspects configured with concrete values. For both cases, a policy model or language needs to be identified to represent these policies. From the obligation type policy domain, two distinct policy languages were identified: Ponder [38][39] and Rei [40].

Ponder [38][39] is a policy language developed by Imperial College for policy-based management systems. Ponder policy language has been used for more than a decade and is well received by the research community. Ponder defines an obligation policy as a policy which is triggered by an event, and must satisfy its conditions before performing the action specified. Ponder policies also have a subject and a target to specify the agent and the objects on which the actions are to be performed. Furthermore, the Ponder obligation policy model has an exception property to specify what action to take in case an error occurs during the policy execution.

Policy language Rei [40] is based on deontic concepts and grounded in semantic language RDF-S [28]. It is not tied to any specific application and permits domain specific information to be added without modification. Rei has constructs for authorisation, prohibition, dispensation and specifically obligation policy rules. Its obligation policy rules are defined as actions that an entity must perform and are usually triggered when a certain set of conditions are true. It consists of actions, conditions, subjects, and policy objects.

In particular, the Rei policy's action is represented as a tuple with action name, target objects, preconditions, and effects. Where the action name is an identifier, target objects are a list of objects on which the action applies, preconditions are conditions that need to be true before the action is performed, and effects are the results of the action performed. Rei policy's action also includes four action operators that can be used to specify complex actions. These action operators are: sequence, where action A and B must be performed in sequence; non-deterministic, where either A or B can be performed but not both; repetition, where action A can be executed several times; and once, where action A can only be performed once. And Rei policy's conditions are based on properties of entities and other domain conditions. Rei allows complex conditions to be built from the operators AND, OR, and NOT.

Both policy languages are powerful and complex, but when comparing them, commonality in their policy model started to emerge. These commonalities are the essential properties of the obligation policy model. It can be seen that both policy languages have a condition property that specifies conditions, which must be met before the policy is executed. They also have an action property that specifies actions that must be performed when the policy is executed. Furthermore, Ponder and Rei have a subject and a target property to specify the agent and the objects on which the actions are to be performed.

Ponder has a trigger action that specifies events that will trigger the policy. However Rei doesn't have this property, instead Rei considers an event as another condition. But it is felt that it is important to differentiate an event from other conditions. Therefore, the trigger property is considered as a valuable property of the obligation policy model.

The policy model to be used must contain these essential properties identified. This policy model will be used for specifying high-level policies for managing adaptive web services, and for creating low-level policies for managing these adaptive services.

2.6 A Survey of Policy Refinement Approaches

Policy refinement is the process of decomposing high-level abstract policies into low-level concrete policies, where these low-level policies realise the intention of their high-level policies [50]. Automated Policy refinement has been the focus of several research projects and initiatives [51]. Policy refinement is needed if policies are to be successfully used to control the internal behaviours of adaptive web services, especially in the situations where policies are used to manage adaptive composite services. Composite web services are made up of several other web services, which perform the intended tasks. Thus, composite services can be considered to be abstract. Policies can be specified to a composite service but must be refined before being applied to the relevant constituent services. If policies are to be used successfully, they need to be specified as high-level policies to manage a composite service, and be refined into low-level policies to enforce the correct behaviours in the constituent elemental services. This section presents a survey of different policy refinement approaches.

One type of approach is to use machine-learning techniques to support policy refinement. Verma [57] presents a case-based reasoning approach to policy refinement where the system maintains a database of past cases. A combination of expert knowledge and observation of the managed system is used to build a database containing the relationship between the policy goals and the configurable parameters of the system. When a goal needs to be achieved, the case database is consulted to find the closest matching case. Otherwise, an interpolation is performed between a set of cases to find a solution.

Different mathematical techniques are suggested to improve the efficiency and accuracy of the refinement process. Such techniques as data pre-processing are used to improve the usefulness of the case database by removing noise in the data as well as irrelevant or redundant data. Furthermore, Gaussian distribution and density function are used together with K-nearest neighbour clustering technique in order to improve the accuracy of the point assignment.

Verma's approach to policy refinement is relatively easy to implement since it does not require a system description or a domain model. Instead, the system needs to have its case database populated with a combination of system configuration parameters used, and the policy goals achieved. However, this task needs to be performed by policy experts with full knowledge of the system. This approach was demonstrated for managing network QoS management [58], but it should be scalable to other domains, such as managing adaptive web services by modifying the database to accommodate web service configurations.

Verma's approach is considered to be semi-automated in refining policies. This approach can only be fully automated if all the possible cases have being accounted for and populated into the database. Thus, the effectiveness of this approach depends on having a rich set of cases. At bootstrap time, the system has no case history. So the system has to be populated with a set of cases observed in pre-operational tests. This approach requires large volumes of data of the operations performed to be accumulated. The case database would normally accumulate measurements of various parameters of a system over a long period of time. The learning time needed for this approach can be an issue for ad-hoc type services such as in Ubiquitous environments.

Furthermore, the complexity of Verma's approach depends on the correctness of its cases; hence it needs a policy expert with domain expertise to manually populate its tables with the correct values. In this approach, the refinement process is only as good as the policy experts, who could accidently introduce errors when manually populating the database.

Another type of approach, suggested by Russo [52], uses a goal elaboration based approach to semi-automated policy refinement. This approach uses goal elaboration to derive a set of sub-goals that satisfies the specified policy goal, and adductive reasoning to create a strategy for policy refinement.

Russo's policy refinement approach uses KAOS [54], goal elaboration method to refine abstract goals into lower level goals by breaking a goal into sub-goals using logically proven refinement patterns. KAOS is a formal technique for elaborating goals grounded in temporal logic, and contains domain specific and domain independent elaboration patterns. Each high-level goal is refined into sub-goals, forming a goal refinement hierarchy using AND & OR operators.

In order to perform the next step of the policy refinement process, a domain hierarchy description for the managed objects must be provided, in addition to a system description as state charts, which is transformed into a formal notation based on Event Calculus [55]. Event Calculus is used to formalise the descriptions before being used for the policy refinement process [56].

Abduction reasoning is then used to infer the strategies that will achieve the elaborated goals. Finally, each refined goal is assigned to managed objects or specific operations. The event and conditions of the high-level policy are mapped, by the user, into the low-level policy.

Russo's policy refinement approach only provides a partial automated policy refinement, because this policy refinement process is automated only for the goals previously refined. The automation of this approach is realised by manually refining a goal using goal elaboration and adductive reasoning and saving the policy refinement strategy of this goal for future refinements. One of the difficulties with Russo's approach is that the policy refinement uses patterns that need to be discovered and mapped, and until they are mapped, automated policy refinement cannot be achieved.

Russo's approach to policy refinement was only demonstrated for managing network QoS management [53]. But it can be envisaged that by creating different policy refinement patterns, it is possible to use this policy refinement approach to manage adaptive web services. However, such a task is manual, and can be complex and tedious. Thus, the complexity of this approach lies in creating these policy refinement patterns, where a policy expert with full knowledge of the domain in question needs to be present.

A goal-oriented policy refinement approach, based on (goal-oriented) requirements engineering and model checking technique, is presented by Rubio-Loyola [60]. It uses KAOS goal elaboration methods to refine goals to lower-level goals, forming a goal refinement hierarchy using AND & OR operators. It then uses linear temporal logic formulae and model checking capabilities to obtain system execution traces aimed at fulfilling low-level goals [59]. Policy information is then extracted from the chosen system execution trace, and finally low-level goals are encoded into refined policies specified in Ponder.

Model checking is performed using a tool called SPIN [63]. It makes use of a system behaviour description specified, as PROMELA, i.e. as state machines that communicate via message passing or shared variables. SPIN not only verifies the properties but also produces execution traces through system simulation. This approach is not automated, and requires administrators to analyse the execution traces and choose appropriate sets of refined policies.

Cassasa-Mont [70] outlines a policy-authoring environment that provides a policy toolkit, called POWER, which refines policies specified. Power toolkit uses policy refinement templates that define the relationship between abstract actions and low-level concrete ones. Domain experts must first develop a set of policy templates, expressed as Prolog programs [75].

The POWER tool has an integrated inference engine that interprets these templates (Prolog programs), to guide the user in selecting the appropriate elements from the management information model to be included in the final policy. The policy-authoring tool makes use of a policy template library, information and system model component, and a device-mapper component for the policy refinement. The policy template library contains a collection of policy templates created by domain experts.

Cassasa-Mont's approach could be used for managing adaptive web services but the difficulty is that someone has to actually implement the policy refinement templates, and these templates can be complex and they need to be implemented by a Prolog expert.

In [3] Kiel presents an automated policy refinement approach which uses Ontology and web service composition. Managed devices/components must have a web service-based interface that is described using OWL-S. This web service description, describes how the component can be managed, i.e. a low-level policy. Policy refinement is achieved through a web service composition. Conditions and actions are extracted from a specified policy, and used as inputs in the web service composer. The web service composer has a matchmaking engine that tries to find the necessary service combination to satisfy these inputs. If the web service composer is successful in generating a composite service, then refined policies can be extracted from the sequence of services.

This approach to policy refinement is demonstrated for the network management domain. However, the transition of this approach to adaptive web service management should be small, since it requires that the managed devices be modelled as web services. It is envisaged that the adaptive behaviours of the web services needs to be modelled as web services in order to use this policy refinement approach.

Guerrero [2] presents a generic ontology-based policy refinement approach that also provides interoperability between high-level and low-level policies. Thus, this refinement approach enables bidirectional policy mapping at runtime. Guerrero's approach needs that high level and low-level of a system be modelled as Ontology using OWL together with the OWL relationship between these Ontology models. Translation rules modelled in Semantic Web Rule Language (SWRL) are needed to allow the data interchanged between the different levels. Policies are modelled as SWRL and refined using the Ontology models, and their relationships.

Another approach, from Albuquerque [64], to policy refinement designed for security type policies, uses a structured technique that models network security systems based on the concepts of policy-based management and model-based management. It describes a security system using a model that specifies the system in different abstraction levels, and policy hierarchies built from the low-level up. Automated policy refinement is achieved using a modelling technique where a system's model is structured in different abstraction levels. System's objects, relationships, and policies at a certain abstraction level together with system model of the lower level and the relationship between entities of the two layers enables the generation of lower level policies.

Systems to be managed and their policies are modelled with several abstract layers, each with objects and associations. The model entities of a certain level and their relationships supply the contextual information needed to automatically interpret and refine the policies of the same level. It uses a 'Diagram of Abstract Subsystem' to model a management system (abstractly) segmented into Abstract Subsystems (AS); a graph comprised of AS.

Another approach to automated policy refinement of security policies is from Cunningham [66]. It models the resource hierarchy that is used to refine policies assigned to the abstract resources in (top of) the resource hierarchy, and automatically produces low-level policies for its concrete resources. Policies are first refined for a resource type, and then for its instance. It uses AND/OR Graph to model resources and Arithmetic and Logical Expression Tree (ALET) to write expressions model policy specification.

Table 1 presents a comparison of the approaches discussed. It focuses on comparing the key aspects of the policy refinement process.

	Russo	Verma	Power	Klie	Cunningham	Rubio-Loyola
Approach Kind	KAOS goal elaboration & abductive reasoning	Case base reasoning & clustering techniques	Policy template	Web Service Composition	Inference of abstract hierarchy model	KAOS goal elaboration
Automated	Semi-automated	Semi-Automated	Not automated	Automated	Automated	Not automated
Policy Type	Goal	Configuration	Management	Obligation	Authorisation	Goal
Domain	Network QoS	Network QoS	Network QoS	Network Management	Network Security	Network QoS
User Expertise Needed	System and Policy Experts	Policy or System configuration Expert	Policy (Prolog) and System experts	System expert	System expert	System and Policy experts
Models used in Refinement	System behaviour as Event Calculus, domain hierarchy, KAOS patterns	Case database	Policy templates (Prolog)	OWL-S Model	DAOG & ALET	FSM, KAOS patterns
User Inputs	State charts, domain hierarchy, KAOS patterns	Case entries	Policy templates	OWL-S Model	Resource type hierarchy	System behaviour
Outputs	Ponder policies	Configuration parameters	Configuration parameters	Configuration parameters	Policy as ALET	Ponder
Key Limitations	Need to cache patterns before being automated	Need to be pre populated with use cases	Need to create Prolog based templates	Need to have all the required web services available for composition	Need a detailed resource type hierarchy	Need to run simulation and manually select valid traces
Complexity	Creating KAOS patterns	Creating use case entries	Creating the policy template	Describing low-level device policies as OWL-S web service	Describing the Resource type hierarchy	Analysing and choosing traces
Tool Support	Yes	Yes	Yes	Yes	Yes	Yes

Table 1, Policy refinement approaches comparison

After surveying the different approaches to policy refinement, some positive and negative points were noted. In order to manage adaptive composite web services, policy refinement is needed. But this policy refinement needs to be automated and easy to use, so that service providers (or service users) can manage these adaptive web services with ease.

Neither Cassasa-Mont's approach, nor Rubio-Loyola's approach are automated, which makes them difficult to use. Russo's approach is only automated if all the policy refinement patterns are present, which initially is not the case, thus a policy expert is required to manually create them. Verma's approach also suffers from initial inability to automate policy refinement, where a policy expert is required to manually populate its database with policy refinement cases.

Cassasa-Mont's approach makes a good point in that it is better to first ensure a valid specified high-level policy by guiding the user, rather than trying to validate refined policies, such as in Rubio-Loyola's approach. A full system description as state charts is needed in both Russo's and Rubio-Loyola's approaches, which can be cumbersome and tedious. It can also lead to state explosion if dealing with large and complex systems.

2.7 Conclusion

This chapter introduced and discussed technologies and techniques in the area of web service technologies, policy-based management systems and modelling languages, which will be used as part of the discussion in the following chapters of this thesis. A survey and analysis was also presented which examines approaches to policy refinement. Finally, a choice of obligation policy technologies used for the proposed solution was presented. The next chapter identifies the assumptions regarding and requirements for the proposed solution to achieving adaptive web services. It also introduces a new methodology for achieving adaptive web services and describes the architectural design of a set of integrated tools to support this methodology.

3 Design of the MAWS Methodology and Requirements for Supporting Tools

3.1 Introduction

An outline of the Manage Adaptive Web Service (MAWS) methodology is presented in this chapter. The MAWS methodology consists of three key development processes namely:

i. to semantically describe adaptive web services;

ii. to describe the composition of adaptive composite web services and to aggregate their adaptive behaviours;

iii. to specify high-level policies for managing these adaptive composite web services and to refine these policies into policies that enforce these adaptive behaviours.

The chapter first presents the definitions and assumptions which scope the design of the proposed methodology. The input and output artefacts for each stage of the methodology are identified and the scope of the tool set which would support this methodology is defined. An example of an adaptive composite service is described which will subsequently be used throughout the chapter to illustrate the proposed methodology. When presenting the MAWS methodology, descriptions of example artefacts produced at each stage of the proposed methodology are also shown. This chapter then presents a requirement analysis which identifies the architectural requirements necessary for the design of a set of integrated tools to support the proposed methodology.

3.2 Definitions and Assumptions

Key definitions, assumptions, and design decisions were made regarding adaptive web services, their description technologies, and policy management. The key assumptions made are:

i. Adaptive behaviours within a web service may change the effect of a service but must not change the input parameters required by the service or the type of output produced by the service (i.e. no change to input and output types).

This thesis considers adaptive web services to be those which have internal adaptive behaviours. These behaviours affect the constituent web services internally without changing their inputs and outputs[6]. Typically, these adaptive behaviours are based on non-functional characteristics, e.g. use a faster search algorithm, or allocate more resources such as bandwidth for higher quality video streaming. However, it is possible that they can change the service's effect, such as, an adaptive behaviour for an email service that encrypts its email contents or compresses its email attachment file. Consequently, adaptive web services have the same inputs and outputs, but may differ in non-functional characteristics, e.g. time, cost, quality, or calculation algorithms. Thus, this thesis focuses on those web services which have adaptive behaviours but do not change the service's inputs and outputs types. This assumption, however, limits the range of services to which the proposed approach can apply.

ii. The adaptive behaviours within web services should be semantically described without any conflict or ambiguity, and this description omits a description of the service's overall (default) behaviour.

Adaptive web services discussed in this thesis have an overall behaviour or default behaviour in the same manner any other web service would have. This default behaviour is not to be semantically described. Instead, only the adaptive behaviours internal to these web services which the developer wishes to expose are to be described semantically. Hence, smaller semantic description models would be created reducing the risk of over complex semantic descriptions. Furthermore, the semantic description representing these adaptive behaviours must be free of any conflict or ambiguity. Although this assumption restricts the proposed approach to only handle services with appropriate semantic descriptions, it does have the benefit of removing the extra complexity needed for handling such services with incomplete descriptions.

iii. Any adaptive behaviour is a composable adaptive behaviour and adaptive behaviours of a composite web service are the resultant aggregation of adaptive behaviours belonging to its constituent services.

---

[6] Semantic web services consider a web service to have inputs, outputs, preconditions, and effects.

It is also assumed that any adaptive behaviour is considered composable adaptive behaviour. Therefore, when aggregating adaptive behaviours for composite web services, the adaptive behaviours are never filtered out. Furthermore, adaptive behaviours of a composite web service are the resultant aggregation of adaptive behaviours pertaining to its constituent services. This means that adaptive behaviours cannot be directly added to (top level) composite web services. But composite web services can expose the adaptive behaviours of their constituent adaptive web services.

iv. Obligation Policies managing the adaptive behaviours of a web service should not conflict with policies managing the adaptive behaviours of other web services.

This thesis focuses on obligation type policies to manage adaptive web services. While it might be possible to use other type of policies such as authentication policies with the proposed approach, the proposed approach was designed with the intention to only use obligation type policies. Furthermore, policy refinement should percolate policies down and not peer to peer where these refined policies are responsible for managing the adaptive behaviours of a web service, and not for, say, the composition of a composed web service. Since adaptive behaviours are internal to web services and web services operate atomically, then by design it is assumed that policies managing the adaptive behaviours of a particular web service should not conflict with the policies managing the adaptive behaviours of another web service, i.e. that there are no conflictions or sharing of resources between web services.

3.3 Example

In order to demonstrate what is meant by adaptive services and their composition, and how the adaptivities of web services are controlled, an example of a PhotoAlbumPrint service comprising a photo processing service called PhotoService, a photo album maker service called PhotoAlbumService, and a print service called PrintService are illustrated in Figure 3-1.

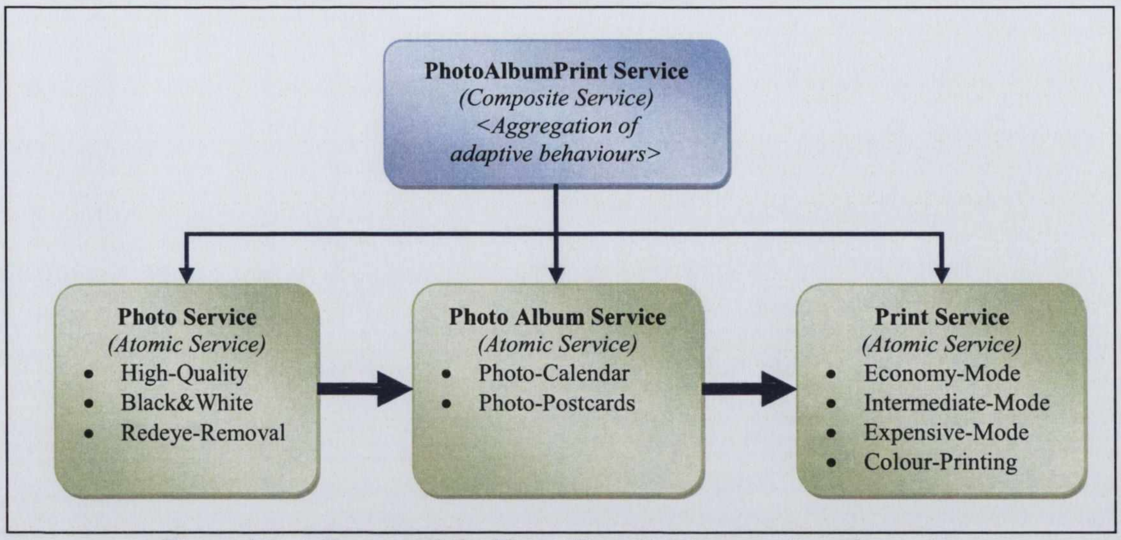


Figure 3-1, A diagram of the PhotoAlbumPrint composite web service

The Photo Service is a photo processing service which processes a given set of photos. This service has the following adaptive behaviours:

- High-Quality adaptive behaviour, which allows the service to process photos in high quality;
- Black&White adaptive behaviour, which modifies the service to process photos in black and white;
- Redeye-Removal adaptive behaviour, which activates a function to remove redeye from photographs.

The PhotoAlbum Service takes a set of photos as input and makes a photo album (as power-point file) from them. It also has adaptive behaviours:

- Photo-Calendar adaptive behaviour, which modifies the service to create a photo calendar;
- Photo-Postcard adaptive behaviour, which modifies the service to create a collection of postcards from the photos.

The Print Service prints any given file, in this case a power-point file. The adaptive behaviours of this service are:

- Economy-mode adaptive behaviour, which modifies the service to print 4 pages of a document per sheet;
- Intermediate-mode adaptive behaviour, which modifies the service to print double-sided;

- Expensive-mode adaptive behaviour, which modifies the service to print single-sided;
- Colour-printing adaptive behaviour, which modifies the service to print in colour if colour cartage is available.

Thus, the PhotoAlbumPrint Service processes a given set of photos, generates a photo album of such photos and prints this album. This service inherits all the adaptive behaviours of its constituent services.

The following high-level policy was specified for the PhotoAlbumPrint Service as an example of a typical management policy used to demonstrate the use of these adaptive behaviours (shown in Table 2):

"When processing portrait photos, apply the Redeye-Removal and use the Create-Calendar adaptive behaviours"

PhotoAlbumPrint Service (CompositeProcess)			
Name	**Event**	**Condition**	**Action**
Policy1	ProcessEvent	Photo= = Portrait	Redeye-Removal && Create-Calendar

Table 2, Management policy for PhotoAlbumPrint Service

This service example will be used in the remainder of this chapter to illustrate the different artefacts produced by the proposed methodology. This service has been kept simple for the sake of clarity and emphasis has been put on highlighting the novel aspects. Full details of the artefacts produced for this example can be seen in Appendix H.

3.4 Overall Design Approach

Before introducing the proposed methodology, some design decisions for the artefacts used in this methodology are presented. Two distinct areas that need addressing are: (i) artefacts for representing adaptive web services and (ii) for representing management policies, before introducing the novel automated policy refinement approach.

Representing Adaptive Composite Web Services

When it comes to defining web services, WSDL [32] is the *de facto* industry standard. Web service containers use WSDL to describe their web services. WSDL describes the syntactical aspects of web services. Web service containers provide WSDL descriptions to their web services so that users or client applications can reason about how to access the required web service. However, it lacks rich semantic definition for describing web services.

From the different modelling languages identified as candidates for representing adaptive web services, OWL-S [36] stood out as the most promising. Besides having the definitions to describe web services semantically, it also has the semantics to describe the composition of composite web services. Furthermore, the semantic definitions of OWL-S are Ontology based and its semantic definitions of web services are grounded in WSDL. Although BPEL [34] can describe web services semantically and the workflow of composite services just as well as OWL-S, BPEL description is based on WSDL and therefore lacks the Ontology-based semantics that OWL-S possess.

It is even more important to have a rich semantic representation of the internal adaptive behaviours of web services. Because the policy-based management used in the proposed approach will need to reason specifically about these adaptive behaviours. It is then vital that a formal representation will be used to describe the adaptive behaviours of adaptive web services. It is possible to attempt to extend OWL-S schema with Ontology definitions to describe internal adaptive behaviours. But it was decided that such an action on its own would be premature, because this extension still needs to have a formal definition for describing adaptive behaviours in a formal manner.

On the other hand, FSM [71] provides a formal representation for describing behaviours of components and systems. This modelling language has been created for formally describing system behaviours and successfully used for many years. By providing a formal representation to these adaptive behaviours, policies can then be created for controlling them. Adaptive web services can then be managed by these policies, which dynamically control the web service's internal adaptive behaviours.

Although FSM has issues with scalability, this issue should not be such a problem when describing adaptive behaviours. The reason is that FSM has not been used to describe the overall behaviour of web services, but only to describe these adaptive behaviours which are encapsulated within web services.

However, FSM was not designed with the intention to describe web services specifically. Therefore, it was decided that FSM should be combined with other modelling languages such as OWL-S in order to better represent adaptive web services. The idea to combine OWL-S with FSM allows composite web services to be described with a modelling language already designed for describing them, and use FSM to describe the adaptive behaviours within these web services.

Representing Management Policies

A policy ontology modelled in OWL was created for obligation type policies based on the properties identified from policy languages Ponder [38][39] and Rei [40]. This Obligation Policy Ontology model will be used by the proposed tools to define high-level management policies, low-level mapping policies, and low-level enforceable policies. The difference between the various policies is their level of abstraction and the context of their parameters, and not the constructs used by the Obligation Policy Ontology model. Policies specified using this model could then be translated to Ponder, Rei or any other rule-based language, thus having the benefit of using any policy engine for this approach.

In this obligation policy ontology model, a policy rule has a *trigger* property that specifies an event that will trigger (activate) the policy. Obligation policies also have a *subject* property that specifies the agents to which the policies applies and which interpret the policies. The *target* property of an obligation policy specifies the object (service) on which the actions are to be performed.

This Obligation Policy Ontology model also contains *condition* property that specify what conditions must be met before the policy is executed. They must be evaluated every time an obligation policy is triggered. The *action* property of an obligation policy specifies the action that must be performed when the policy is executed. This action consists of method invocations for a concrete component or an action name for an abstract component making the policy a high-level (abstract) policy.

Automated Policy Refinement

A policy refinement method is necessary in order to manage the adaptive behaviours of composite services using a policy-based management system. A survey of automated policy refinement approaches was conducted, and presented in section 2.6. Although none of them were designed to manage adaptive web services, they did provide valuable information on what is needed for an automated policy refinement method for managing adaptive web services.

Firstly, the policy refinement needs to be automated since the process is very complex and tedious for any user. It was decided to use an approach that infers its refinement decision based on a rich semantic model of the adaptive web service. The semantic model must also provide vocabulary for specifying appropriate high-level policies. By allowing users to only specify appropriate high-level policies, it is possible to ensure that the refinement of these policies is achievable. Lastly, tools must be provided to facilitate the creation of the semantic models as well as the specification of the high-level policies and their automated policy refinement.

3.5 The MAWS Methodology

The Manage Adaptive Web Service (MAWS) methodology is a design process for describing adaptive (composite) web services semantically, and for specifying and refining management policies to manage the internal adaptive behaviours of these adaptive web services.

Before deploying adaptive web services with enforceable policies to manage their adaptive behaviours, these web services and their adaptive behaviours must first be semantically described. When adaptive web services are composed of other web services, i.e. composite services, their composition must be described and the adaptive behaviours belonging to their constituent services aggregated. Then high-level policies can be specified to manage these adaptive composite services. The specification of these management policies must be based on the semantic representation of adaptive web services. These policies can then be automatically refined (using the semantic descriptions) into low-level policies which are generated to manage adaptive composite services. The policy refinement method is based on the semantic description of adaptive web services. By limiting the policy specification to the vocabulary provided by these semantic descriptions, it is possible to assure that the policy refinement is achievable and that the refined policies are correctly refined according to the intention of the specified high-level policy.

The MAWS methodology presented in this section is designed to guide developers in describing adaptive web services and to allow managers to control these adaptive web services by specifying and refining management policies to enforce their adaptive behaviours. The MAWS methodology defines the required tasks to be accomplished before being able to deploy and execute these adaptive composite services, which results in the creation of enforceable policies to dynamically manage their adaptive behaviours at run-time. These policies allow adaptive (composite) services to change their behaviours in order to:

i. personalise web services according to user's preference;

ii. change web services according to a new business model;

iii. allow web services to be context-aware to changes in the environment

The MAWS methodology was inspired by the methodology used for describing and composing non-adaptive web services [31] and can be divided into three processes, namely Web Service Description, Web Service Composition, and Web Service Management. These are shown graphically in Figure 3-2.

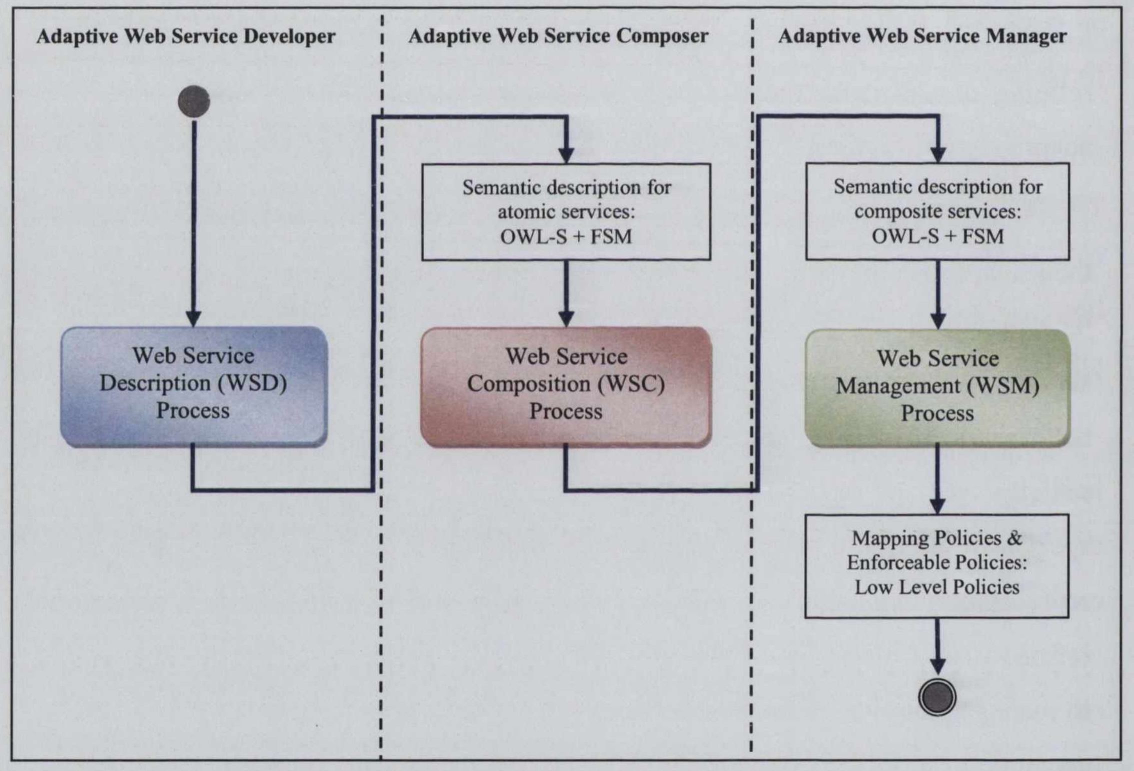

Adaptive Web Service Developer
Adaptive Web Service Composer
Adaptive Web Service Manager
Semantic description for atomic services: OWL-S + FSM
Semantic description for composite services: OWL-S + FSM
Web Service Description (WSD) Process
Web Service Composition (WSC) Process
Web Service Management (WSM) Process
Mapping Policies & Enforceable Policies: Low Level Policies

Figure 3-2, The three processes in the MAWS Methodology

The initial step of the MAWS methodology is the Web Service Description (WSD) process, which is responsible for describing adaptive atomic services. It produces syntactic descriptions for atomic web services (as WSDL), semantic descriptions for atomic web service (as OWL-S), and formal representations (expressed in FSM) for each of the adaptive behaviours pertaining to these adaptive atomic services.

The Web Service Composition (WSC) process is the second step in the MAWS methodology. This WSC process is responsible for modelling and describing the composition of web services, thus producing a semantic description for composite web services (expressed in OWL-S). The WSC process is also responsible for the aggregation of the adaptive behaviours of the constituent web services. This produces formal representations (expressed in FSM) of the adaptive behaviours of the adaptive composite services. The WSC process uses artefacts produced by the WSD process to semantically describe adaptive composite services.

The final step in the MAWS methodology is the Web Service Management (WSM) process, which enables high-level policies to be specified to manage adaptive (composite) services. This process is also responsible for the automated refinement of these policies and the generation of enforceable (low-level) policies for the constituent web services. In addition, it performs a validation of these generated enforced (low-level) policies against the specified (high-level) management policies. The activities in this process are accomplished with the aid of the artefacts produced from the previous process, e.g. OWL-S and FSM.

The MAWS methodology tries to support different kinds of users involved in the development of managing adaptive web services, and encourages that each artefact is produced by the appropriate person responsible for it. Hence, adaptive web services are described by developers who are responsible for creating them. Adaptive web services can then be combined to create adaptive composite services with their aggregated adaptive behaviours. And the composition of these amalgamated web services is described by web service composers or application developers. Finally, adaptive composite services can have their adaptive behaviours managed by policies which can be safely specified by service managers and assured that refined enforceable policies are generated to manage them.

Each of the steps in the MAWS methodology described above consists of a number of design activities. To better understand these processes and their activities, a detailed description of each of them, together with their produced artefacts, is provided in the following subsections. Full details of these artefacts are available in Appendix H.

3.5.1 Web Service Description Process

The Web Service Description (WSD) process was designed to describe adaptive atomic services, both syntactically, as WSDL, and semantically, as OWL-S. Furthermore, it also semantically describes the adaptive behaviours internal to atomic services, as FSM. This is an important step in the MAWS methodology, since these descriptions will be used later to specify management policies and to automatically refine them.

The WSD process is composed of three activities; Web Service Syntactical Description, Web Service Semantic Description and Web Service Adaptivity Description, as depicted in Figure 3-3.

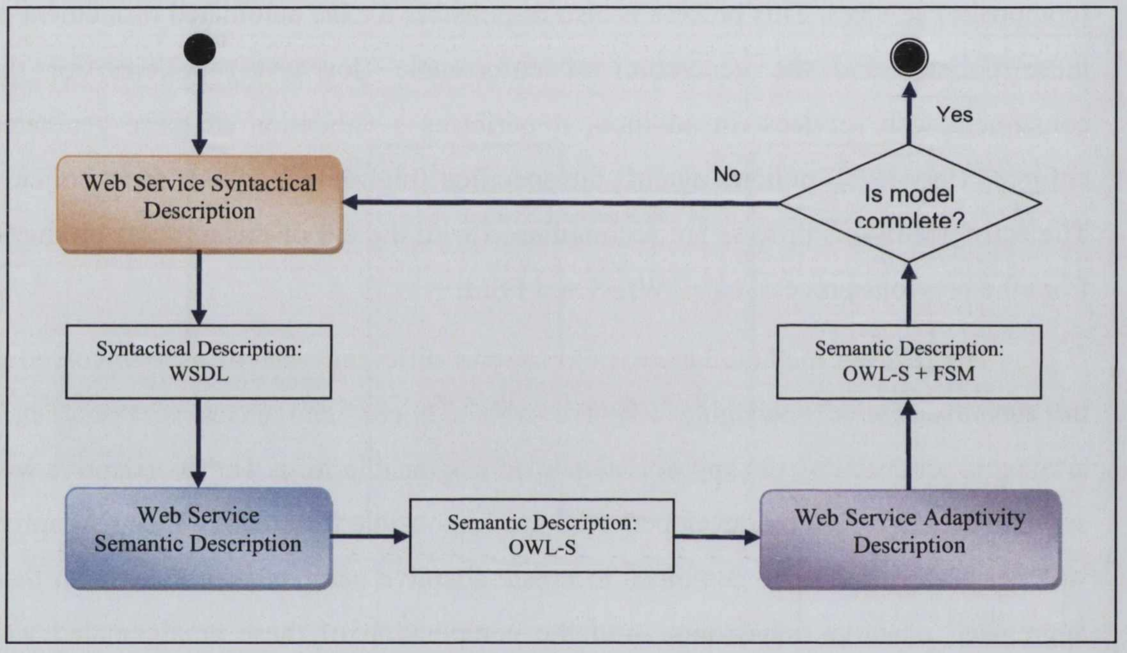

Web Service Syntactical Description
Syntactical Description: WSDL
Web Service Semantic Description
Semantic Description: OWL-S
Web Service Adaptivity Description
Semantic Description: OWL-S + FSM
Is model complete?
No
Yes

Figure 3-3, Web service description process

The WSD process cycle starts with the Web Service Syntactical Description activity, where a given web service is syntactically described as WDSL. The next activity is the Web Service Semantic Description activity, where a given web service is semantically described using an ontology language to produce a web service description in OWL-S. The Web Service Adaptivity Description activity then describes the adaptive behaviours of a web service, producing a formal representation for them as FSMs.

The Web Service Syntactical Description activity provides a syntactical description for web services. This activity assumes that developers have designed their web services, and have either implemented or are about to implement these web services. This activity describes atomic web services as WSDL. A snippet example of the WSDL description is shown in Figure 3-4, which was produced by this activity for a web service example PrintService. It shows a WSDL description of the web service's operation – print; web service's input parameters: document (doc), pages, and colours; web service's response: sheets.

```
<wsdl:definitions targetNamespace="http://print.services">
<wsdl:documentation>PrintService</wsdl:documentation>
<wsdl:types>
<xs:schema attributeFormDefault="qualified"
elementFormDefault="qualified"
targetNamespace="http://print.services/xsd">
...
<wsdl:message name="printMessage">
<wsdl:part name="part1" element="ns0:doc"/>
<wsdl:part name="part2" element="ns0:pages"/>
<wsdl:part name="part3" element="ns0:colours"/>
</wsdl:message>
<wsdl:message name="printResponse">
<wsdl:part name="part1" element="ns0:sheets"/>
</wsdl:message>
<wsdl:portType name="PrintServicePortType">
<wsdl:operation name="print">
<wsdl:input message="axis2:printMessage"
wsaw:Action="urn:print"/>
<wsdl:output message="axis2:printResponse"/>
</wsdl:operation></wsdl:portType>
...
```

Figure 3-4, Snippet of WSDL artefact consumed for print web service

The Web Service Semantic Description activity provides a semantic description for web services. This activity consumes the syntactic description, as WSDL, of a web service and enriches it with a semantic description. It describes web services semantically with the use of OWL-S [36]. OWL-S semantically describes web services through three aspects: *process*, *profile*, and *grounding*, which are grouped under a fourth aspect called *service*. For this thesis emphasis was given to the *service, process* and *grounding* aspects of the OWL-S model. This activity produces a semantic description for web services as OWL-S, which describes web service's type, their inputs and their outputs, as well as, a mapping to their WSDL description.

Figure 3-5 shows a snippet of the artefact produced by this activity for the web service PrintService. It shows the *service* and *process* aspects of the OWL-S description for the PrintService. The *service* aspect describes the location of the other aspects of the service semantic description. The code snippet of the *process* aspect describes the PrintService's inputs: document name, number of pages, number of colours to be used; and outputs: number of sheets used to print the document.

```
…
<service:Service rdf:ID="PrintService">
   <!-- Reference to the Process Model -->
   <service:describedBy rdf:resource="&gprocess;#PrintProcess"/>
   <!-- Reference to the Grounding -->
   <service:supports
rdf:resource="&ggrounding;#PrintServiceGrounding"/>
   <!-- Reference to the Profile -->
   <service:presents
rdf:resource="&gprofile;#PrintServiceProfile"/>
</service:Service>
…
  <process:AtomicProcess rdf:ID="PrintProcess">
    <process:hasInput rdf:resource="#DocumentName"/>
    <process:hasInput rdf:resource="#NumberOfPages"/>
    <process:hasInput rdf:resource="#NumberOfColours"/>
    <process:hasOutput rdf:resource="#NumberOfSheets"/>
    <process:hasFiniteStateMachine
rdf:resource="PrintProcessFSM.owl#PrintProcessFSM"/>
  </process:AtomicProcess>
…
```

Figure 3-5, Snippet of OWL-S artefact produced for print web service

The next step is to add an adaptive behaviour description for atomic web services. The Web Service Adaptivity Description activity describes the internal adaptive behaviours of adaptive atomic services. This activity enriches web service semantic descriptions (expressed in OWL-S), provided by the previous activity, with a formal representation of the web service's adaptive behaviours, expressed as FSM. FSM uses its events, transitions, and states to describe these adaptive behaviours semantically.

This activity should not be used to describe the service's default behaviour but only the behaviours that are desired to be exposed which modify the service's behaviour. In order to contain the description of several adaptive behaviours in a generic manner, the FSM description starts with a description of the different running states of a web service. These states are: idle, input, process, and output states. FSM representing adaptive behaviours are then associated to these states through *submachine* property. In this manner adaptive behaviours can be grouped based on the service's running states.

The print web service example has four adaptive behaviours:

- ExpensiveMode, which modifies the print service to print single-sided;

- IntermediateMode, which modifies the print service to print double-sided;
- EconomyMode, which modifies the print service to print two pages per sheet double-sided (i.e. four pages per sheet);
- ColourPrinting, which modifies the print service to print in colour;

Figure 3-6 shows a snippet of the FSM (modelled in OWL) describing a simple adaptive behaviour called EconomyMode. This snippet shows two states: initial state (EconomyModeInitialState) and final state (EconomyModeState1), as well as a transition from initial state to final state contained in this FSM. The description of the final state details the activity to be performed by this state – to change printing mode to print 2 pages on a side and print double sided. This FSM describing the adaptive behaviour EconomyMode is contained within a submachine, see snippet code below, and associated to the service's FSM via a sub-FSM.

```
<fsm:SubmachineState rdf:ID="EconomyModeSubSM">
  <fsm:submachine>
    <fsm:StateMachine rdf:ID="EconomyMode">
      <fsm:top>
        <fsm:CompositeState rdf:ID="EconomyModeCS">
          <fsm:subvertex>
            <fsm:PseudoState rdf:ID="EconomyModeInitialState">
              <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
              <fsm:outgoing>
       <fsm:Transition rdf:ID="EconomyModeInitialStateInitialTransition">
                  <fsm:trigger>
                    <fsm:SignalEvent rdf:resource="InitialEvent"/>
                  </fsm:trigger>
                  <fsm:source rdf:resource="#EconomyModeInitialState"/>
                  <fsm:target rdf:resource="#EconomyModeState1"/>
                </fsm:Transition>
              </fsm:outgoing>
            </fsm:PseudoState>
          </fsm:subvertex>
          <fsm:subvertex>
            <fsm:FinalState rdf:ID="EconomyModeState1">
              <fsm:doActivity>printmode(2, doublesided)</fsm:doActivity>
 <fsm:incoming rdf:resource="#EconomyModeInitialStateInitialTransition"/>
            </fsm:FinalState>
          </fsm:subvertex>
        </fsm:CompositeState>
      </fsm:top>
      <fsm:comment>prints 4 pages per sheet</fsm:comment>
    </fsm:StateMachine>
  </fsm:submachine>
</fsm:SubmachineState>
```

Figure 3-6, FSM artefact describing the adaptive behaviours of the print web service

3.5.2 Web Service Composition Process

The Web Service Composition (WSC) process was designed to describe the composition of adaptive composite services, and to aggregate their adaptive behaviours. This process describes the adaptive behaviours of adaptive composite services by aggregating the adaptive behaviours, previously described as FSM, of the constituent atomic web services. This step in the MAWS methodology is important because adaptive web services being managed can either be atomic or composed. For those services that are composed, it is necessary to describe their composition. The composition description will be used later by the policy refinement process to automatically refine high-level policies. Furthermore, the aggregation of adaptive behaviours will allow them to be exposed to service managers specifying high-level policies for composite services, as seen in the next MAWS process.

The WSC process is composed of three activities: Web Service Composition Modelling, Web Service Composition Description, and Web Service Adaptivity Aggregation, as depicted in Figure 3-7.

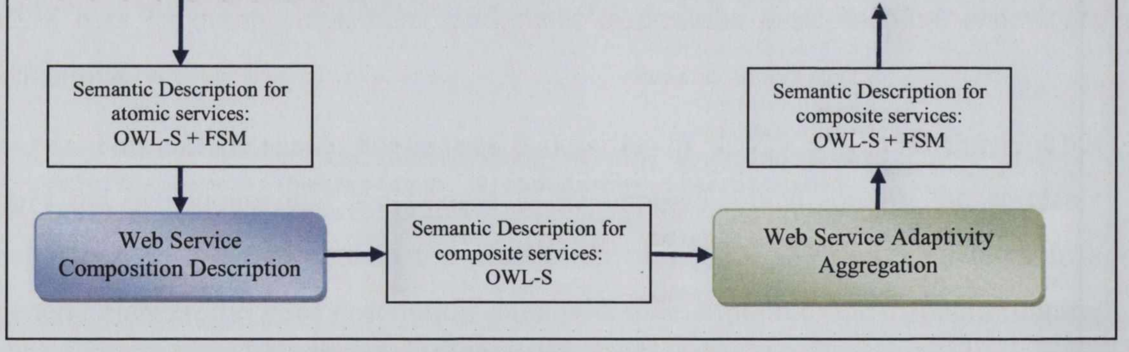

Web Service Composition Modelling
No
Is model complete?
Yes
Semantic Description for atomic services: OWL-S + FSM
Semantic Description for composite services: OWL-S + FSM
Web Service Composition Description
Semantic Description for composite services: OWL-S
Web Service Adaptivity Aggregation

Figure 3-7, Web service composition process

The WSC process cycle starts with the Web Service Composition Modelling activity which models composite services either manually or in an automated manner. Then the Web Service Composition Description activity describes the composition of the modelled composite web service. Lastly, the Web Service Adaptivity Aggregation activity aggregates the adaptive behaviours belonging to the constituent web services for the given composite web service. It is assumed that the web services used in the composition have undergone the WSD process.

The web services to be deployed can be either atomic or composite web services. However, since composite web services are composed of atomic or other composite services, their composition must be modelled. The purpose of this process is to describe the composition of adaptive composite services.

The Web Service Composition Modelling activity searches for the available web services to satisfy the set of composition requirements, it then selects the appropriate candidate web services for the composite web service, and finally it models the composite service. This activity can be performed manually or in an automated manner, such as through the use of an AI planner [5]. The tasks of this activity are research topics in themselves and outside the scope of this thesis. This activity provides a point of reference to developers performing the composition of web services before describing the composition of composite web services in the next activity.

The Web Service Composition Description activity describes composite web services semantically, producing semantic descriptions for composite web services as OWL-S. The semantic description of composite web services is similar to that of atomic services, except for the addition to a description of their composition. The composition is described in a hierarchical manner by identifying the constituent services and describing their control flow, as well as the dataflow between the inputs and outputs of the constituent services. For example, this activity can describe the composition of the PhotoAlbumPrint web service, as OWL-S, exhibited in the snippet depicted in Figure 3-8. This snippet details the control flow of the composite service as a sequence of three services: PhotoService, PhotoAlbumService, and PrintService.

```
  <process:CompositeProcess rdf:ID="PhotoAlbumPrintProcess">
    <process:composedOf>
      <process:Sequence>
        <process:components>
          <process:ControlConstructList>
            <objList:first rdf:resource="#PerformPhotoService"/>
            <objList:rest>
              <process:ControlConstructList>
                <objList:first>
       <process:Perform rdf:resource="PerformPhotoAlbumService">
                </objList:first>
                <objList:rest>
                  <process:ControlConstructList>
            <objList:first rdf:resource="#PerformPrintService"/>
          <objList:rest rdf:resource="generic/ObjectList.owl#nil"/>
                  </process:ControlConstructList>
                </objList:rest>
              </process:ControlConstructList>
            </objList:rest>
          </process:ControlConstructList>
        </process:components>
      </process:Sequence>
    </process:composedOf>
  </process:CompositeProcess>
...
```

Figure 3-8, OWL-S snippet describing the composition of the PhotoAlbumPrint web service

Once a composite web service is described, its adaptive behaviour must also be described. The adaptive behaviours of a composite web service are the resultant aggregation of the adaptive behaviours belonging to the constituent adaptive services. Thus, the main role of the Web Service Adaptivity Aggregation activity is to combine the adaptive behaviours pertaining to the constituent adaptive services of a given composite service. The aggregation activity aggregates adaptive behaviours in a reversed hierarchical manner based on the composite description of a given composite web service, provided by the previous WSCM activity. The resultant aggregated adaptive behaviours for a composite web service are described as FSM in the same manner adaptive behaviours are described for atomic web services.

The output of this activity for the example composite web service PhotoAlbumPrint is shown in Figure 3-9. It depicts a FSM snippet (modelled in OWL) describing the aggregated adaptive behaviours of this composite web service. The snippet indicates adaptive behaviours: CreateCalendar adaptive behaviour, which creates a photo calendar document with a given set of photos and originated from AlbumService; and ExpensiveMode, which prints a document page per sheet and originated from PrintService.

```
<fsm:CompositeState rdf:ID="ProcessState">
  <fsm:subvertex>
    <fsm:SubmachineState rdf:ID="CreateCalendarSubSM">
      <fsm:submachine>
        <fsm:StateMachine rdf:ID="CreateCalendar">
        …
          <fsm:comment>
            Creates a calendar with the given set of photos
          </fsm:comment>
        </fsm:StateMachine>
      </fsm:submachine>
    </fsm:SubmachineState>
  </fsm:subvertex>
  <fsm:subvertex>
    <fsm:SubmachineState rdf:ID="ExpensiveModeSubSM">
      <fsm:submachine>
        <fsm:StateMachine rdf:ID="ExpensiveMode">
        …
          <fsm:comment>prints a page per sheet</fsm:comment>
        </fsm:StateMachine>
      </fsm:submachine>
    </fsm:SubmachineState>
  </fsm:subvertex>
</fsm:CompositeState>
```

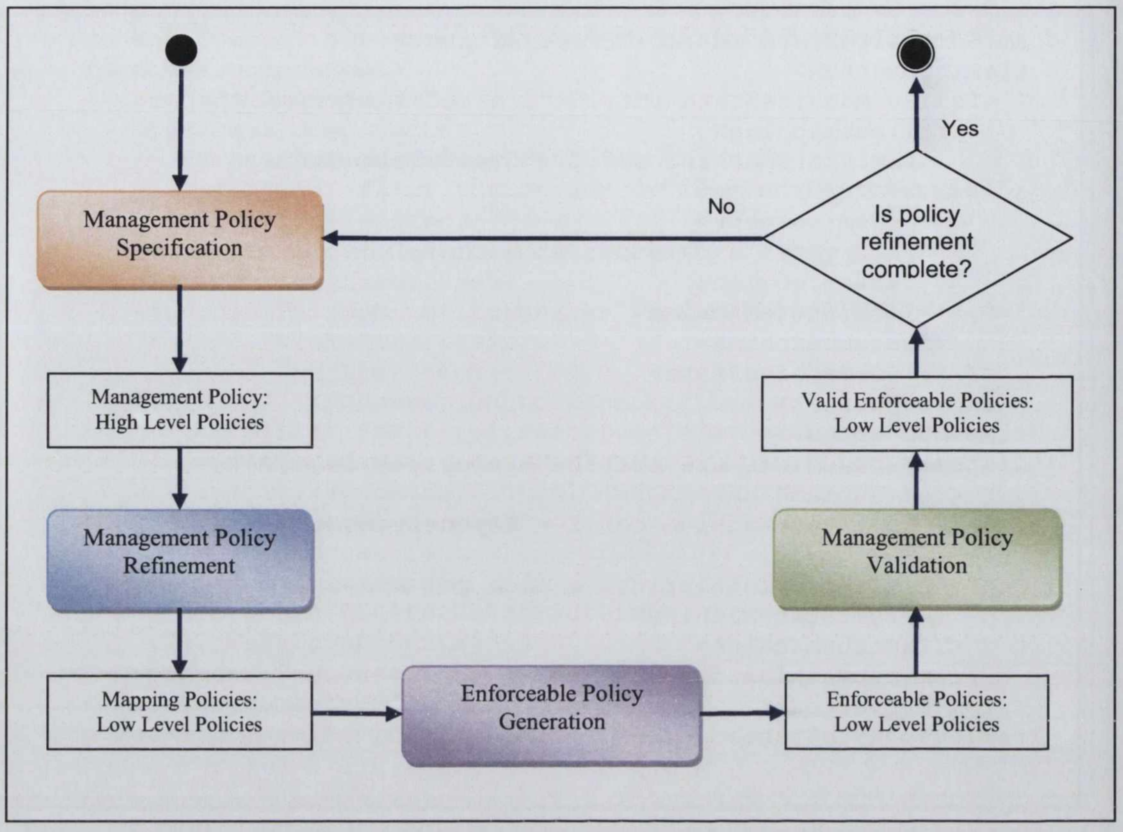
Figure 3-9, FSM snippet for PhotoAlbumPrint service's adaptive behaviours

At this level, the system is operating under the assumption that any adaptive behaviour is a possible adaptive behaviour for a composite service. Hence, the system is merely providing all the possible behaviours of the constituent services, and it is not trying to provide any filtering of combination that might be inappropriate i.e. conflicting adaptive behaviours with same name.

3.5.3 Web Service Management Process

The Web Service Management (WSM) process was designed to allow web service managers to specify high-level (obligation type) policies for managing adaptive (composite) services, and have these policies refined, generating enforceable low-level policies. These low-level policies are then validated before being used at run-time to dynamically manage adaptive web services.

The WSM process is composed of four activities: Management Policy Specification, Management Policy Refinement, Enforceable Policy Generation, and Management Policy Validation, as depicted in Figure 3-10.



Yes
No
Management Policy Specification
Is policy refinement complete?
Management Policy: High Level Policies
Valid Enforceable Policies: Low Level Policies
Management Policy Refinement
Management Policy Validation
Mapping Policies: Low Level Policies
Enforceable Policy Generation
Enforceable Policies: Low Level Policies

Figure 3-10, Web service management process

The WSM process cycle starts with the Management Policy Specification activity, which specifies the high-level policies needed to manage adaptive (composite) services. The Management Policy Refinement activity then refines these high-level management policies into *mapping policies* which are disseminated down the service's composition to the relevant constituent atomic services. Subsequently, the Enforceable Policy Generation activity generates low-level *enforceable policies* for the relevant constituent atomic services, which are enforced on these services to manage their adaptive behaviours. Lastly, these enforceable policies are validated by the Management Policy Validation activity to ensure there are no conflicting policies.

The Management Policy Specification activity specifies the management policies to manage the adaptive behaviours of composite web services. These management policies are high-level obligation type policies. The semantic descriptions for web service (OWL-S) and their adaptive behaviours (FSM), which was produced by the activities within the WSC process, are used as vocabulary to specify management policies. The expression of these management policies is limited to the terms of the semantic descriptions provided. This activity ensures that only functionally valid management policies can be specified.

An example of a management policy for managing the PhotoAlbumPrint service is depicted in Figure 3-11. This policy was specified to change the default behaviour of the PhotoAlbumPrint service to use RemoveRedEye (functionality for removing redeye from people in photographs) and CreateCalendar adaptive behaviour when processing portrait photos. This policy is described in the Obligation Policy Ontology model (see Appendix E), expressed in OWL. Details of this policy are shown in the code below, depicting the use of the named adaptive behaviours as its action.

```
<policy:Policy rdf:ID="PhotoAlbumPrintPolicy1">
 <rdfs:comment>Policy for this service to use removeredeye and
create calendar adaptive behaviours for processing portrait
photos</rdfs:comment>
 <policy:target rdf:resource=
"PhotoAlbumPrintProcess.owl#PhotoAlbumPrintProcess"/>
<policy:event>
 <policy:SimpleEvent rdf:ID="PhotoAlbumPrintPolicy1Event1"/>
</policy:event>
<policy:condition>
<policy:SimpleCondition
rdf:ID="#PhotoAlbumPrintPolicy1Condition1">
  <policy:subject rdf:resource="#CameraPhotoCategory"/>
  <policy:predicate>
   <policy:Predicate rdf:ID="equal"/>
   </policy:predicate>
  <policy:value>portrait</policy:value>
</policy:SimpleCondition>
</policy:condition>
<policy:action>
<policy:ComplexAction rdf:ID="PhotoAlbumPrintPolicy1Action1">
<rdfs:first> <policy:AndList> <rdfs:first>
 <policy:SimpleAction rdf:ID="PhotoAlbumPrintPolicy1Action101">
  <policy:value>RemoveRedEye</policy:value>
</rdfs:first><rdfs:rest>
  <policy:SimpleAction rdf:ID="PhotoAlbumPrintPolicy1Action102">
   <policy:value>CreateCalendar</policy:value>
  </policy:SimpleAction>
 </rdfs:rest> </policy:AndList> </rdfs:first>
 <rdfs:rest></rdfs:rest>
</policy:ComplexAction> </policy:action> </policy:Policy>
```

Figure 3-11, Management policy specified for PhotoAlbumPrint web service

The objective of the Management Policy Refinement activity is to refine the management policies specified for managing adaptive composite services. These policies are refined according to the web service composition generating refined policies, called mapping policies. These mapping policies, as the name suggests, map high-level management policies to low-level enforceable policies (which are produced by the next activity). These mapping policies are assigned to the relevant constituent services to accomplish the policy refinement. This activity uses artefacts from the

WSC process to perform policy refinement and specifically for the generation of mapping policies; artefacts used include web service syntactical description (WSDL), web service semantic description (OWL-S), and adaptive behaviour's formal representation (FSM).

In order to satisfy the condition aspect of the high-level policy, specified for the PhotoAlbumPrint Service, refined mapping policies must first be generated to notify the (future) enforceable policy targeting the PhotoService to monitor its input for value "portrait". Furthermore, refined mapping policies also need to be generated targeting the relevant services to notify that such condition occurred. These mapping policies trigger (future) enforceable policies that will enforce the desired adaptive behaviours. Figure 3-12 illustrates these two refined mapping policies produced by this activity.

PhotoAlbumPrint Service (CompositeProcess)			
Name	**Event**	**Condition**	**Action**
PhotoAlbum PrintPolicy1	ProcessEvent		event(PhotoAlbumPrintPolicy1Event1C0)
PhotoAlbum PrintPolicy1CU	PhotoAlbumPrint Policy1Event1CU0		event(PhotoAlbumPrintPolicy1Event1aA) && event(PhotoAlbumPrintPolicy1Event1bA)

PhotoAlbumPrint Service (CompositeProcess)			
Name	**Event**	**Condition**	**Action**
PhotoPolicy1C	PhotoAlbumPrint Policy1Event1aA		event(PhotoRemove RedEyeEvent)

PhotoAlbumPrint Service (CompositeProcess)			
Name	**Event**	**Condition**	**Action**
PhotoAlbumPolicy1C	PhotoAlbumPrint Policy1Event1bA		event(PhotoAlbumCreateCalendarEvent)

Figure 3-12, Refined mapping policies for PhotoAlbumPrint composite web service

The Enforceable Policy Generation activity completes the policy refinement process by generating low-level enforceable policies needed to satisfy the high-level management policies specified to manage adaptive composite services. These enforceable policies are generated for the relevant refined mapping policies. They are created to either monitor the inputs and outputs of constituent atomic services or to enforce their adaptive behaviours. The generation of these enforceable policies is based on the semantic and syntactical descriptions of the relevant atomic web service, and the formal description (FSM) of the pertaining adaptive behaviour they are enforcing.

Figure 3-13 shows the refined enforceable policy generated for the constituent web service PhotoService to monitor its input PhotoCategory, and fires an event if this input is of value "portrait". This policy is triggered by `PhotoAlbumPrintPolicy1Event1C0` event and if the condition is met, it generates an event called `PhotoAlbumPrintPolicy1Event1CU0`. Other policies were generated to enforce the adaptive behaviours RemoveRedeye and CreateCalendar.

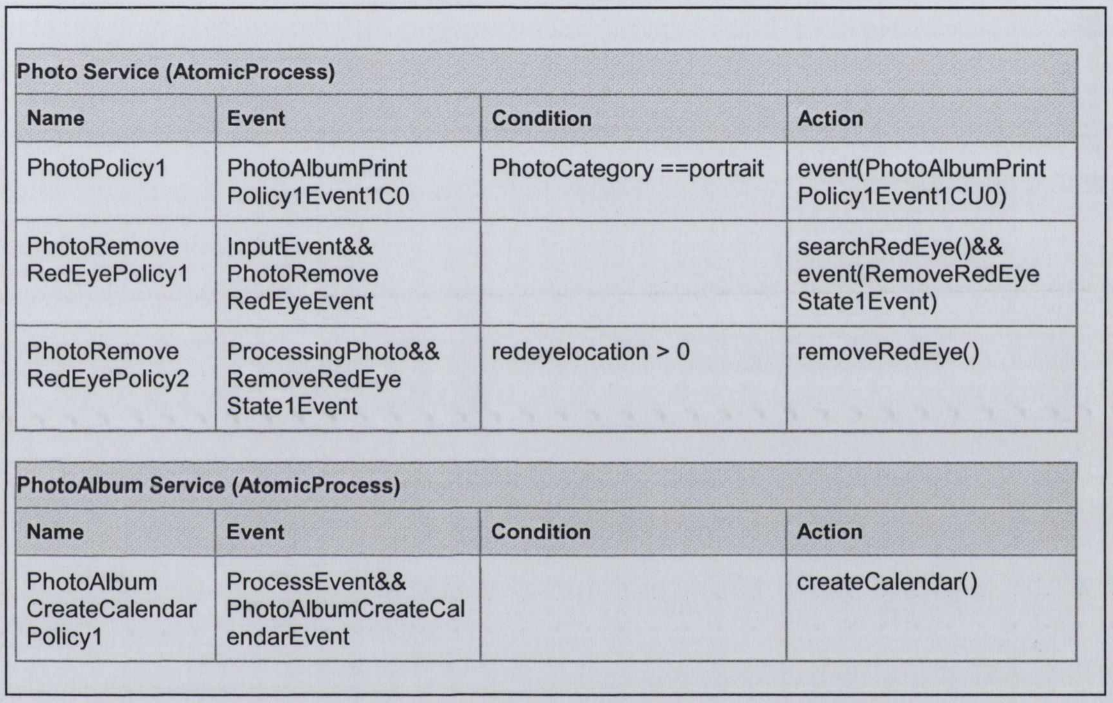


Photo Service (AtomicProcess)			
Name	**Event**	**Condition**	**Action**
PhotoPolicy1	PhotoAlbumPrint Policy1Event1C0	PhotoCategory ==portrait	event(PhotoAlbumPrint Policy1Event1CU0)
PhotoRemove RedEyePolicy1	InputEvent&& PhotoRemove RedEyeEvent		searchRedEye()&& event(RemoveRedEye State1Event)
PhotoRemove RedEyePolicy2	ProcessingPhoto&& RemoveRedEye State1Event	redeyelocation > 0	removeRedEye()

PhotoAlbum Service (AtomicProcess)			
Name	**Event**	**Condition**	**Action**
PhotoAlbum CreateCalendar Policy1	ProcessEvent&& PhotoAlbumCreateCal endarEvent		createCalendar()

Figure 3-13, Refined enforceable policies for the PhotoAlbumPrint web service

Although these activities are tedious, it can be seen later that the Management Policy Refinement and Enforceable Policy Generation activities are performed automatically by the relevant supporting tool. Thus, neither developers nor web service managers need to expend time and effort performing these activities by hand.

Finally, in order to assure that adaptive web services can be deployed safely with their enforceable policies, these policies must first be validated. The last activity in the WSM process is the Management Policy Validation activity, which verifies that the generated refined management policies enforce the behaviours of an adaptive web service without any confliction. Confliction can occur if contradicting policies are specified for the same adaptive service. This activity identifies any policy confliction in order to remove them by re-specifying the management policies, so that there are no conflicting policies among the generated policies. Once these enforceable policies are validated, adaptive web services can be deployed together with these enforceable policies to manage them. Policy confliction is a researched area which is outside the scope of this thesis.

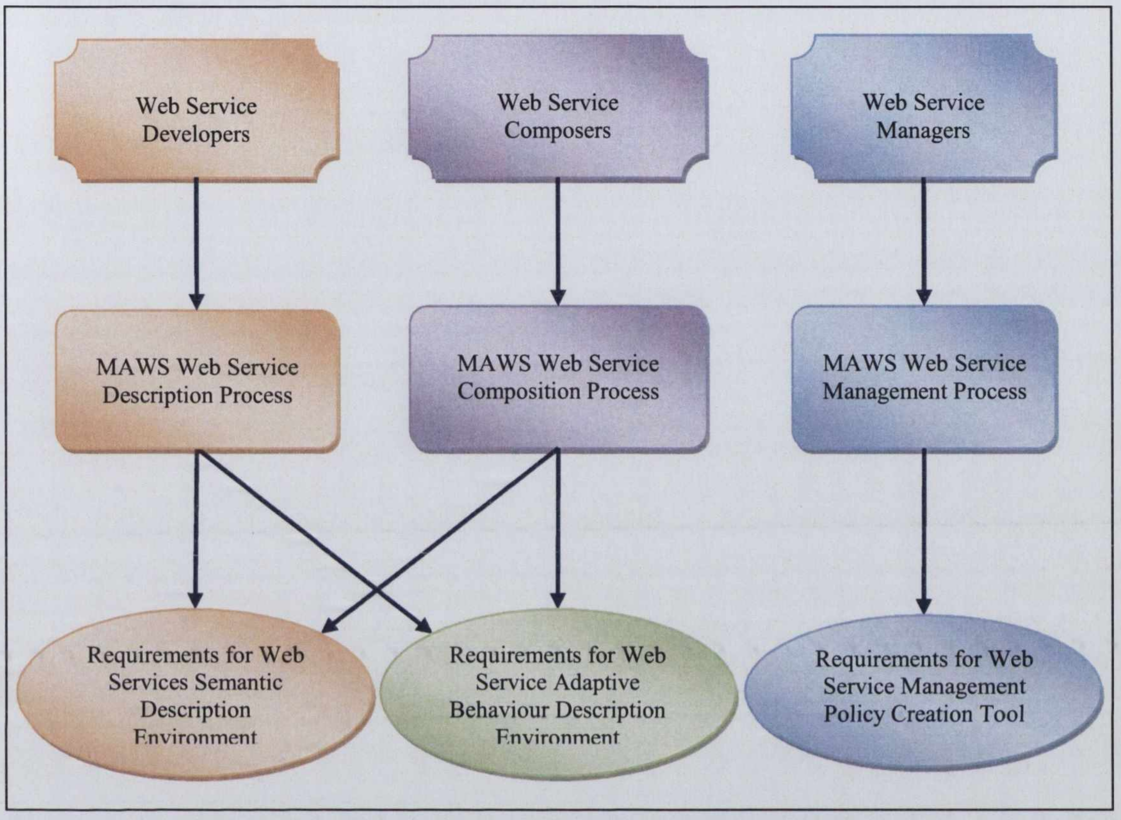
At this level, the system is operating under the assumption that any adaptive behaviour is a possible adaptive behaviour to be used by the management policy. Hence, the system is merely providing all the adaptive behaviours available in the web services for management policy specification, and it is not trying to provide any filtering of combinations that might be inappropriate. Thus, the system could generate enforceable policies that are syntactically correct but semantically invalid. This is a limitation in our approach, which could be addressed in future work by expanding the FSM model with taxonomy or semantic tags, that would allow a semantic filter reasoner to prevent or filter out such inappropriate combinations, during management policy specification.

3.6 Supporting Tools for the MAWS Methodology

It is fair to say that using the MAWS methodology manually would be tedious and complicated, as can be seen from the snippet artefact examples produced by the various activities from the MAWS methodology. In order to use the MAWS methodology successfully, there is a need for supporting tools to accomplish the activities. Before designing these supporting tools, a set of requirements needs to be defined. In reality, there are three different specific users that can be identified which will be using the MAWS methodology, namely web service developers, web service composers, and web services managers.

3.6.1 Requirements for the Supporting Tools

A set of requirements for integrated tools to support the MAWS methodology are outlined in the following sub-sections. These requirements are based on the needs of the various MAWS methodology activities, in addition to the objectives of this thesis and influences from the current state of the art (see Chapter 2). These requirements can be divided into three categories (see Figure 3-14), i.e. those specific to capturing semantic descriptions of web services, those specific to capturing descriptions of web service's adaptive behaviours, and those related to managing adaptive web services using policies. These requirements can also be divided into three different user roles: web service developers, web service composers, and web service managers. Furthermore, a set of secondary requirements concerned with the tool's overall design were also identified.



Web Service Developers
Web Service Composers
Web Service Managers
MAWS Web Service Description Process
MAWS Web Service Composition Process
MAWS Web Service Management Process
Requirements for Web Services Semantic Description Environment
Requirements for Web Service Adaptive Behaviour Description Environment
Requirements for Web Service Management Policy Creation Tool

Figure 3-14, Alignment of requirements with MAWS processes

Requirements for Web Services Semantic Description Environment

The functional requirements for designing a tool to semantically describe web services were identified as:

- The tool should allow users to create semantic descriptions for atomic services as OWL-S – derived from the *Web Service Semantic Description* activity;
- The tool should allow users to describe the composition of composite services, including a description of their control flow and dataflow – derived from the *Web Service Composition Description* activity;

A set of secondary requirements for designing this tool to semantically describe web services are:

- The tool should be able to facilitate users in describing web services semantically by providing a graphical user interface;

- The tool should be able to persistently store the web service's semantic description (OWL-S) in a consistent format as OWL;

Requirements for Web Service's Adaptive Behaviour Description Environment

The functional requirements for designing a tool to semantically describe adaptive behaviours of web services were identified as:

- The tool should allow users to create a FSM model (shell) for web services so as to contain the descriptions of their adaptive behaviours – derived from the *Web Service Adaptivity Description* activity;
- The tool should allow users to create sub-FSMs (child FSM to be contained within FSM shell) representing adaptive behaviours of web services – derived from the *Web Service Adaptivity Description* activity;
- The tool should allow users to describe the adaptive behaviours of web services as FSM, using states, transitions, and events – derived from the *Web Service Adaptivity Description* activity;
- The tool should be able to automatically aggregate the adaptive behaviours of the constituent services of composite services according to their composition – derived from the *Web Service Adaptivity Aggregation* activity;

A set of secondary requirements for designing this tool to describe adaptive web services are:

- The tool should be able to facilitate users in describing web service's adaptive behaviours by providing a graphical user interface;
- The tool should be able to persistently store the semantic description of adaptive behaviours belonging to web services, as FSM, in a consistent format as OWL;

Requirements for Web Service Management Policy Creation Tools

A set of functional requirements for designing a tool to create policies to manage the adaptive behaviours of web services were identified as:

- The tool should allow users to specify high-level management policies to manage the adaptive behaviours of web services – derived from the *Management Policy Specification* activity;
- The tool should be able to express management policies as obligation policies with events, conditions, and actions – derived from the *Management Policy Specification* activity;
- The tool should limit users in specifying management policies to be expressed with the description of the managed adaptive web service – derived from the *Management Policy Specification* activity;
- The tool should provide users, when specifying policies, with the appropriate vocabulary of adaptive behaviours as their action, web service's parameters as their condition – derived from the *Management Policy Specification* activity;
- The tool should be able to automatically refine management policies into mapping policies according to the web service's composition and the description of their adaptive behaviours – derived from the *Management Policy Refinement* activity;
- The tool should be able to automatically generate low-level enforceable policies from the refined mapping policies assigned to constituent adaptive services according to their semantic descriptions – derived from the *Enforceable Policy Generation* activity;

A set of secondary requirements for designing a tool to create management policies to manage the adaptive behaviours of web services were identified as:

- The tool should be able to persistently store the management policies and the enforcement policies in a consistent format;
- The tool should have a graphical user interface that facilitates the user in specifying management policies and that shows details of the web service to be managed, and their adaptive behaviours
- The tool should be able to enable users to specify management policies for managing the adaptive behaviours of web services by preventing them from making syntactical mistakes and helping users when specifying policies by providing the appropriate vocabulary;

3.6.2 Architecture of the Supporting Tools

An overview architectural design is presented for a set of integrated tools needed to support specific activities in the MAWS methodology. The requirements identified above provide key guidance in determining the overall architecture of these tools.

Rather than developing a single application, the preferred choice was to design a set of integrated tools to suit the different categories of users in the development cycle, and to best support specific activities in the MAWS methodology. Figure 3-15 shows a set of integrated tools which was designed and includes:

i. a tool for describing both atomic and composite web services semantically as OWL-S;
ii. a tool for providing a formal representation of the adaptive behaviours of adaptive (composite) services as FSM;
iii. a tool for authoring high-level management policies for adaptive (composite) services and automatically generating refined mapping and enforceable policies to manage the adaptive behaviours of these adaptive composite services.

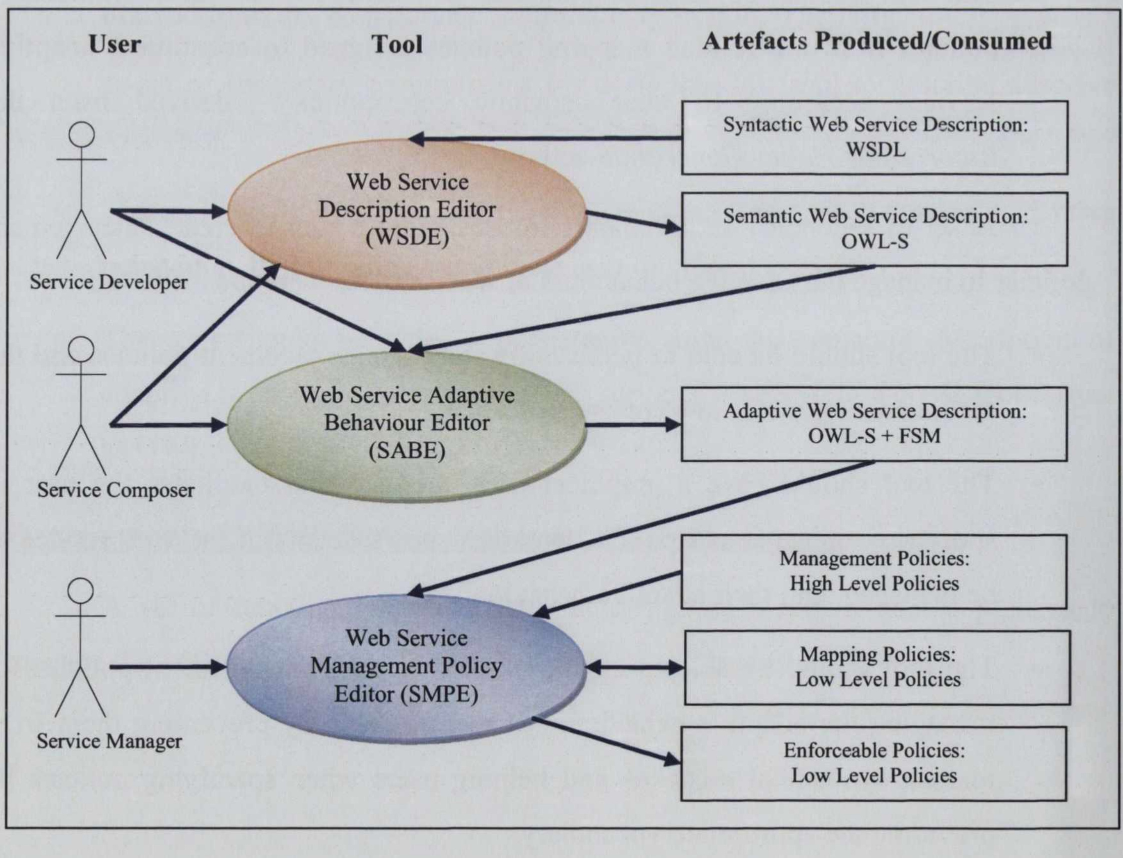

User
Tool
Artefacts Produced/Consumed
Service Developer
Service Composer
Service Manager
Web Service Description Editor (WSDE)
Web Service Adaptive Behaviour Editor (SABE)
Web Service Management Policy Editor (SMPE)
Syntactic Web Service Description: WSDL
Semantic Web Service Description: OWL-S
Adaptive Web Service Description: OWL-S + FSM
Management Policies: High Level Policies
Mapping Policies: Low Level Policies
Enforceable Policies: Low Level Policies

Figure 3-15, Design architecture of the set of integrated tools for supporting MAWS methodology

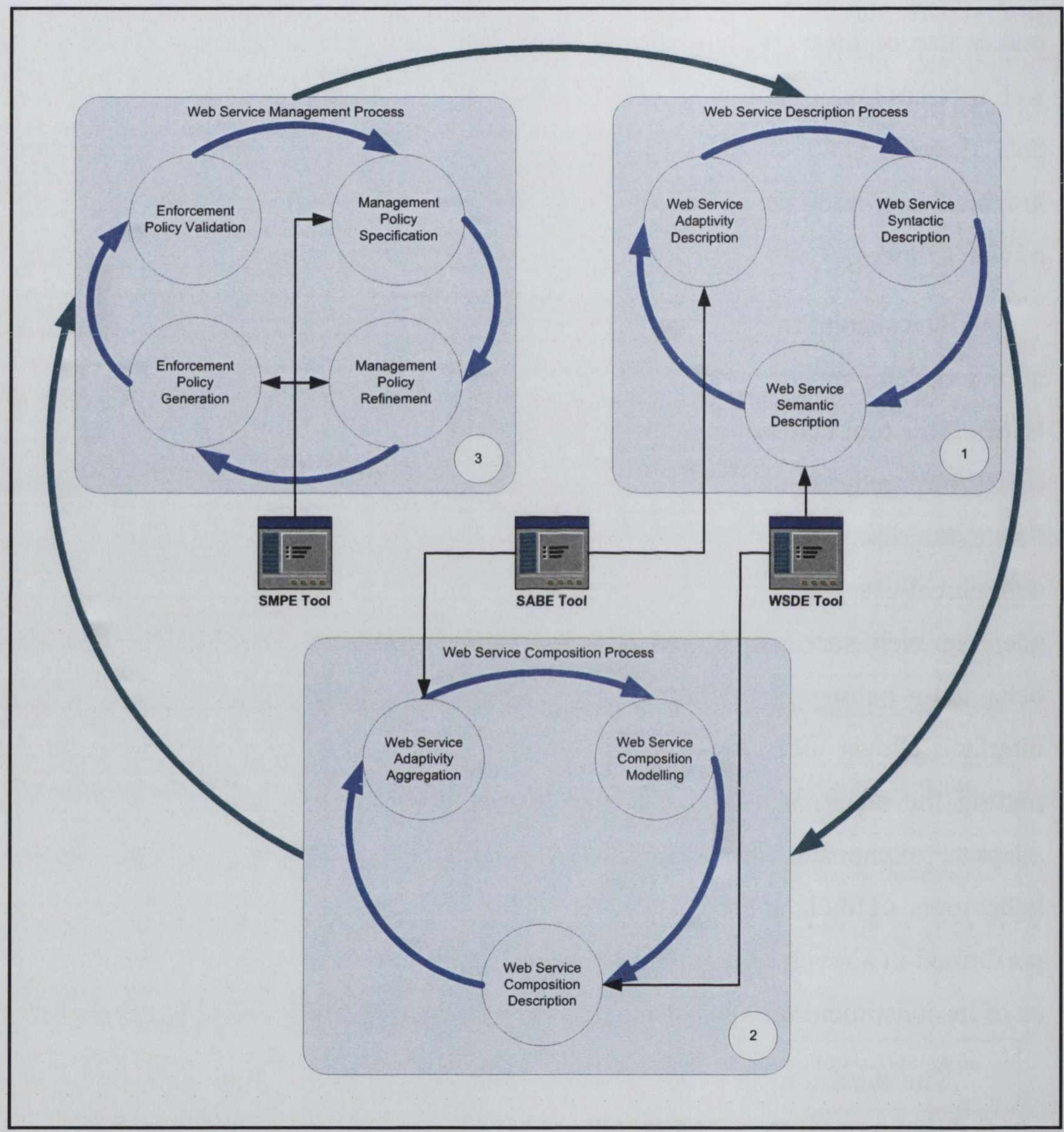
In particular, the first tool is a Web Service Description Editor (WSDE), which was designed with the objective of allowing developers to semantically describe web services as OWL-S. This tool produces OWL-S descriptions for both atomic and composite web services. The WSDE tool expects atomic web services to be already implemented and to have their syntactical aspects described as WSDL and makes use of these WSDL descriptions when producing semantic descriptions for web services i.e. grounding model of OWL-S. This tool describes the control flow and data flow as OWL-S when describing composite services. It is designed with a graphical user interface to facilitate users in viewing and editing semantic descriptions of web services.

The second tool is a Web Service Adaptive Behaviour Editor (SABE), which allows developers to describe the adaptive behaviours of adaptive web services as FSMs. This tool takes as input semantic descriptions of web services as OWL-S. The SABE tool generates a FSM for an adaptive web service which encompasses all the FSMs describing each of the service's adaptive behaviours. This tool is designed with a graphical user interface to facilitate users in describing adaptive behaviours of these adaptive web services. The SABE tool allows users to view and edit adaptive behaviours belonging to these web services as FSM. In addition, the tool's user interface allows users to view the semantic descriptions of web services thereby putting the adaptive behaviours into context with the adaptive web services. For adaptive composite services, this tool automatically aggregates the adaptive behaviours of their constituent adaptive web services. This aggregation process is performed in a reversed hierarchical manner and it produces a FSM, of similar format as of its constituent services, for the adaptive behaviours of a composite service.

The third tool is a Web Service Management Policy Editor (SMPE), which allows users to specify management policies to manage adaptive composite services. The SMPE tool takes as input adaptive web services syntactically described as WSDL, their semantic descriptions as OWL-S, and their adaptive behaviours described as FSM. This tool uses these descriptions as policy vocabulary to express the management policies being specified and for auto-generating refined (mapping and enforceable) policies. The SMPE tool is designed with a graphical user interface to facilitate the user in specifying high-level management policies for adaptive composite services; it also allows users to view the semantic descriptions of adaptive web services.

3.6.3 Supporting the MAWS methodology

Figure 3-16 shows how the supporting tools designed and described in Section 3.6.2 support the various activity of the MAWS methodology.



Web Service Management Process
Enforcement Policy Validation
Management Policy Specification
Enforcement Policy Generation
Management Policy Refinement
3
Web Service Description Process
Web Service Adaptivity Description
Web Service Syntactic Description
Web Service Semantic Description
1
SMPE Tool
SABE Tool
WSDE Tool
Web Service Composition Process
Web Service Adaptivity Aggregation
Web Service Composition Modelling
Web Service Composition Description
2

Figure 3-16, How the designed tools support the MAWS methodology processes.

As described in the MAWS methodology section, see Section 3.4, the first step, WSD, contains three activities, two of which can be supported by the proposed tools. The first activity, Web Service Syntactical Description activity, can be supported by the web service container, which produces WSDL descriptions for web services. The proposed tool WSDE supports the second activity, Web Service

Semantic Description activity, producing an OWL-S description for web services. The Web Service Adaptive Behaviour Description activity is supported by the proposed SABE tool. The SABE tool produces FSM to describe the adaptive behaviours within adaptive atomic services.

The second step in the MAWS methodology, the WSC process, contains three activities, two of which are supported by the designed tools. The Web Service Composition Description activity is supported by the WSDE tool which semantically describes the composition of composite web services as OWL-S. Whereas, the Web Service Adaptivity Aggregation activity is supported by the SABE tool, which automatically aggregates the adaptive behaviours of the constituent adaptive services described as FSM.

The third step in the MAWS methodology, the WSM process, contains four activities, three of which are supported by the designed SMPE tool. The SMPE tool supports the Management Policy Specification activity by allowing users to specify management policies to manage adaptive (composite) services. This tool also supports the Management Policy Refinement activity by automatically refining the specified policies according to the description of the adaptive web service. It produces mapping policies which map the high-level management policies to low-level enforceable policies. Furthermore, this tool supports the Enforcement Policy Generation activity by automatically generating enforceable policies for the constituent adaptive atomic services so as to manage their adaptive behaviours.

3.7 Summary

This chapter has illustrated how the MAWS methodology semantically describes web services and their adaptive behaviours, composing them into composite web services and aggregating their adaptive behaviours, as well as specifying management policies, refining these policies and generating enforceable policies to manage the described adaptive composite services.

A set of identified requirements and an architecture design for a suite of integrated tools to support the MAWS methodology was presented in this chapter. An overall picture of the architecture has been given, focusing on how the various tools support the MAWS methodology described. A set of tools were proposed as supporting tools for the MAWS methodology together with a description of their architecture. A detailed discussion of the implementation of these tools is carried out in the next chapter, Chapter 4.

4 Implementation of Integrated Tools

4.1 Introduction

The MAWS methodology was described in the previous chapter – Chapter 3, and three tools were identified so as to support this methodology. The proposed integrated tools are:

i. **Web Service Semantic Description Editor (WSDE)**, a tool that describes the semantic description of atomic web services and the composition description of composite services expressed as OWL-S;

ii. **Web Service Adaptive Behaviour Editor (SABE)**, a tool that describes the adaptive behaviours belonging to atomic web services and that automates the aggregation of these behaviours for composite services;

iii. **Web Service Management Policy Editor (SMPE)**, a tool responsible for the specification of high-level management policies and for the refinement of these policies and generation of low-level policies to manage these adaptive behaviours.

These integrated tools facilitate web service developers and managers to describe adaptive web services, specify high-level management policies for them, and auto-generate refined low-level enforceable policies to manage these adaptive web services.

This chapter describes the design and implementation of this suite of integrated tools following the requirements identified in the previous chapter. A detailed description is provided of all the main components within each tool, and any issues encountered in the implementation of these tools are highlighted. Next, a step by step demonstration of how to use these integrated tools is presented. Finally, a policy evaluation platform that was implemented to provide the necessary runtime support for evaluating adaptive web services and their management policies during runtime is described.

4.2 Design of Integrated Tools

These tools were designed as a component based architecture with a Graphical User Interface (GUI) and an Ontology interpreter to parse and reason about the semantic descriptions of adaptive web services. The GUI facilitates the users in viewing and editing the semantic descriptions of adaptive web services or in specifying management policies for these adaptive web services.

These tools were implemented solely in Java [76] allowing them to be used over several platforms since Java is a platform independent language. The GUI of these tools was implemented with Swing [44]: a GUI toolkit for Java that provides widgets such as text boxes, buttons, panels, and tables. Swing widgets are designed to be consistent across all platforms and considered more sophisticated GUI components than the earlier Abstract Window Toolkit (AWT) [43].

Jena [46], a toolkit which provides a programmatic environment for RDF, RDFS and OWL was used in the implementation of these tools as the Ontology interpreter. Jena was used for interpreting and reasoning about ontological descriptions expressed in OWL. Jena provided these tools with the ability to read, parse, interpret, and save the semantic descriptions related to adaptive web services and their management policies. By using Jena, these tools can read and write ontology descriptions as OWL instances to file, database, or memory. Reading and writing to a file was the preferred option as it was simple to use and yet provided persistence.

4.3 Design and Implementation of the WSDE tool

The Web Service Description Editor (WSDE) tool allows developers to describe web services semantically, producing an OWL-S description for both atomic and composite web services. This tool was developed to support the SABE and the SMPE tools. The WSDE tool is not novel in describing web service as OWL-S, but it does provide a user interface for describing web services which is consistent with the other tools developed.

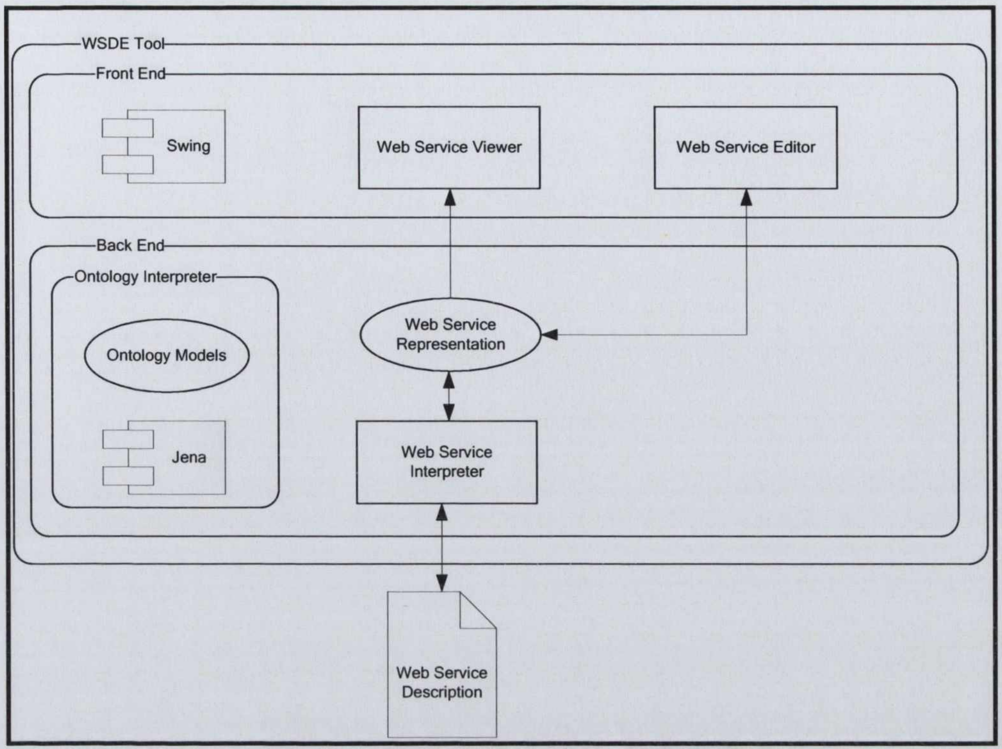


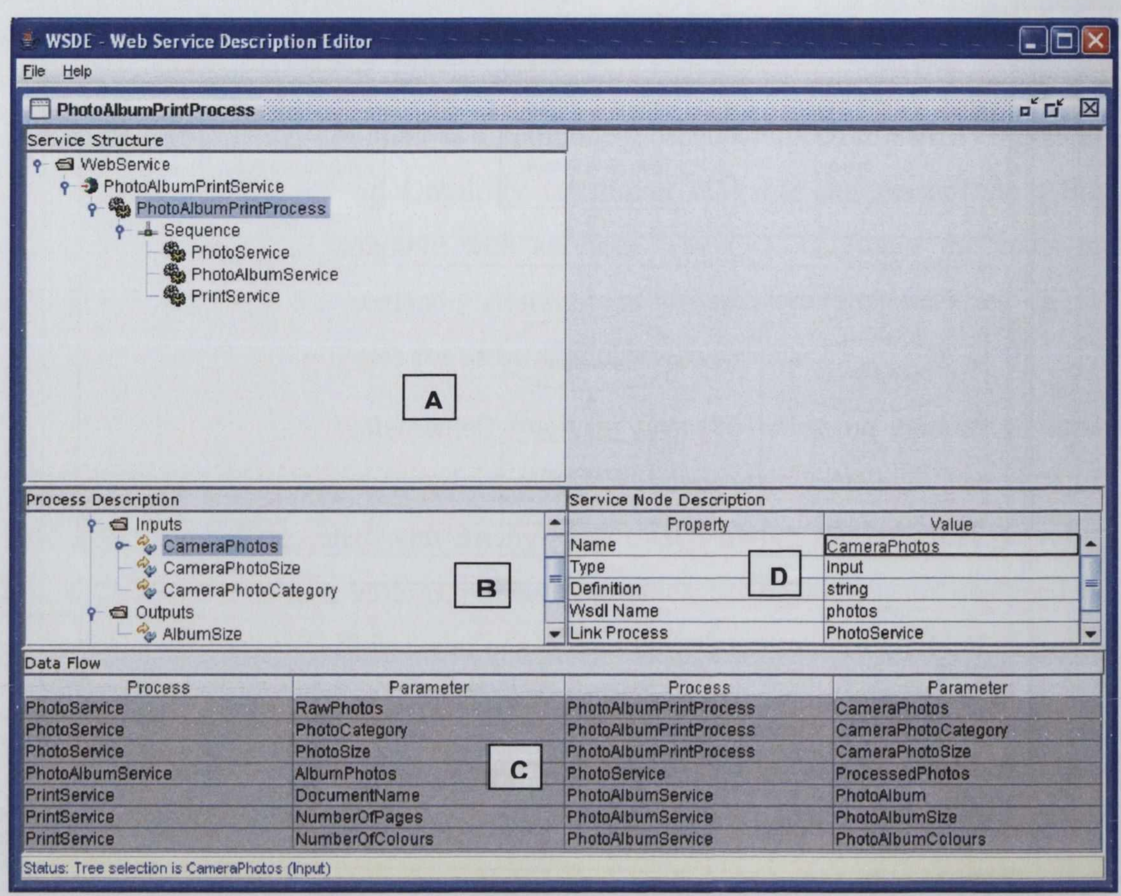
Figure 4-1, Architecture of the WSDE tool

This tool was implemented as a component based architecture, as depicted in Figure 4-1, with the following main components:

- Ontology Interpreter Component, for interpreting and persisting web services descriptions model as OWL-S;
- Web Service Viewer Component, for allowing users to view web service's semantic details;
- Web Service Editor Component, for allowing users to create or edit web services semantic descriptions and their details;

These components are described in detail in the next sub-sections below.

This tool was implemented with a Graphical User Interface (GUI) which helps developers to view and edit the particulars of web service semantic descriptions. This tool allows developers to semantically describe the parameters of web services and the composition of composite web services. In addition, developers can save the web service description as an OWL file. This tool utilises an OWL reasoner to read or save web service's semantic descriptions, specified in OWL-S, to an OWL file in lexical form. A screenshot of the WSDE tool is shown in Figure 4-2.



Service Structure

```
WebService
  PhotoAlbumPrintService
    PhotoAlbumPrintProcess
      Sequence
        PhotoService
        PhotoAlbumService
        PrintService
```

Process Description

```
Inputs
  CameraPhotos
  CameraPhotoSize
  CameraPhotoCategory
Outputs
  AlbumSize
```

Service Node Description

Property	Value
Name	CameraPhotos
Type	Input
Definition	string
Wsdl Name	photos
Link Process	PhotoService

Data Flow

Process	Parameter	Process	Parameter
PhotoService	RawPhotos	PhotoAlbumPrintProcess	CameraPhotos
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	CameraPhotoCategory
PhotoService	PhotoSize	PhotoAlbumPrintProcess	CameraPhotoSize
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos
PrintService	DocumentName	PhotoAlbumService	PhotoAlbum
PrintService	NumberOfPages	PhotoAlbumService	PhotoAlbumSize
PrintService	NumberOfColours	PhotoAlbumService	PhotoAlbumColours

Status: Tree selection is CameraPhotos (Input)

Figure 4-2, A screenshot of the WSDE tool

The following panels are depicted in Figure 4-2 (seen in an anti-clockwise rotation starting with the top left panel):

A. Web service composition detail panel, which displays the composition of composite web services i.e. the control flow of a composite service;

B. Web service process detail panel, which displays the details of a selected service (termed as process in OWL-S);

C. Web service data flow detail panel, which displays the data flow of a composite service;

D. Service node description panel, which allows users to edit the details of selected service nodes.

4.3.1 Ontology Interpreter Component

The Ontology Interpreter component provides the functionality for parsing, interpreting, and saving semantic descriptions of web services based on OWL. This component uses Jena to parse, interpret, and save ontological description of web services (as OWL-S). By using Jena, this component is able to read and write ontology descriptions as OWL instances to an OWL file in lexical form.

Although Jena can read and write ontology descriptions from OWL files in lexical form, it cannot interpret them without an ontology schema and logic code to reason about them. This component was implemented with a sub-component to interpret the Ontology model used by the WSDE tool. This sub-component contains the necessary Ontology schema to interpret and reason about the semantic web service descriptions as OWL-S.

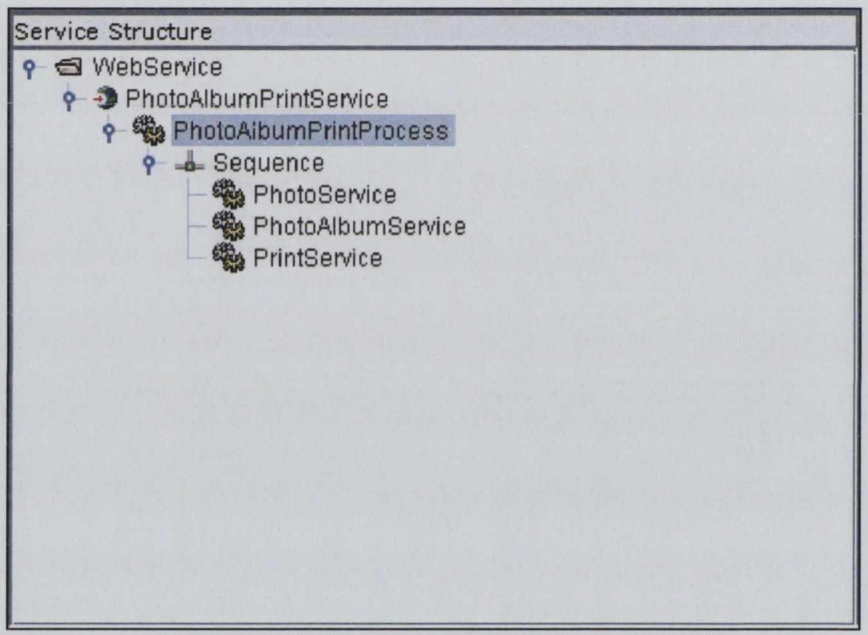
An internal representation of a semantic web service description model was created in order to facilitate the visualisation of the web service description. This component performs better with an internal representation than using information taken directly from the web service Ontology model since the internal representation doesn't require to be interpreted every time it is called. This internal representation of web services is populated by this component once the Ontology model instance is interpreted.

When saving semantic web service descriptions, this sub-component uses the OWL-S Ontology schema to create an instance of OWL-S Ontology model for a particular web service and save it in an OWL file in lexical form. The description of the web service is taken from their internal representation, and necessary OWL-S specific information is included into the Ontology model. In the case of a composite web service, this process saves the web service description as OWL-S Ontology model in a hierarchical manner starting from the topmost web service to every last constituent atomic web service while also including details of their control flow and dataflow.

4.3.2 Web Service Viewer Component

The web service viewer component uses panels to display the details of the web service's semantic description. These panels are Java Swing widgets, which follow the Model View Controller (MVC) architectural pattern [77]. MVC divides the functionality into three parts: model, which handles the model to be displayed; view, which handles the displaying of the model; and controller, which handles the events such as actions from the user. It provides the view aspect of MVC by inheriting from the widget component and extending it. These widgets expect the model and control aspects of MVC to be implemented and provided to them, using their API, before they can be instantiated. The model aspect of the MVC for the swing widgets is implemented as the internal representations of web services. This visual component contains the implementation of the actions for each of the events from the widgets, i.e. the control aspect of MVC.

This component displays the details of web services to the user using three panels. The first panel displays the control flow of a web service in case this service is a composite one. The second panel is a context sensitive panel which displays the details of a service selected from the first panel. The third panel displays the dataflow of a service in case this service is a composite service.


Service Structure

```
WebService
    PhotoAlbumPrintService
        PhotoAlbumPrintProcess
            Sequence
                PhotoService
                PhotoAlbumService
                PrintService
```

Figure 4-3, Web service composition detail panel

The first panel, depicted in Figure 4-3, displays the web service's composition details using a tree widget customised with different icons for services and composition constructs. The first node is a node called "WebService" which can contain nodes representing a web service, e.g. PhotoAlbumPrintService. Based on the OWL-S definition, *process* is used to define the making of a web service, where it describes its type and its parameters [36]. In this case PhotoAlbumPrintProcess is an example of such *process*, of type composite.

Composite services must contain a construct node, and a construct node can contain other construct nodes and service nodes depending on the type of construct node. For example, a sequence construct node can contain several web services such as PhotoService, PhotoAlbumService, and PrintService. However, leaf nodes must be atomic services otherwise the model is incomplete. The nodes in the tree were implemented to have a name which is displayed, and a type which is used to determine what icon should be displayed. For example, a cogs icon is used to represent web services, and a pipe-joint icon is used to represent web service's constructs. An action was implemented for when a service node is selected which causes its details to be loaded in the next panel.

This panel also has a context sensitive property pane for each of the nodes, which allows users to add or remove web service's constructors and web service nodes. Depending on which node is selected, the appropriate action is enabled. For example, if a composite service node is selected, then the available options are to add a constructor and to remove the service node.

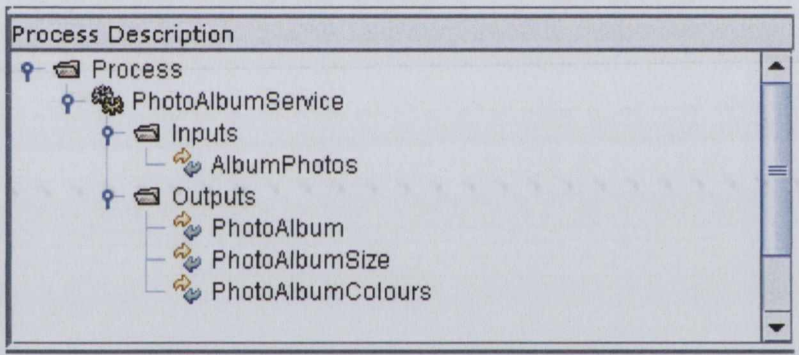
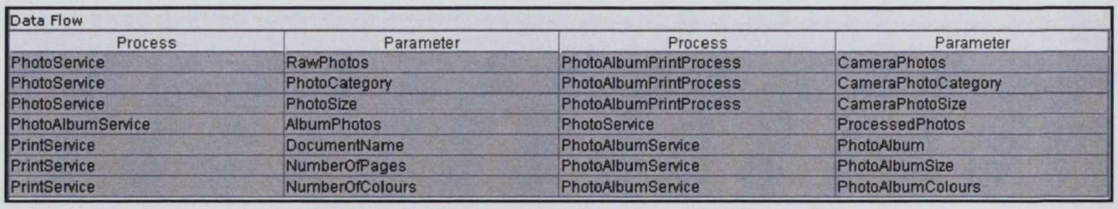


Figure 4-4, Web service process detail panel

The second panel, depicted in Figure 4-4, displays the details of a web service. Its title is "process description" since process is used to define the making of a web service in OWL-S [36]. This panel uses a tree widget customised with different icons, and provides virtual nodes to display, in a categorical manner, the details of a web service, i.e. virtual nodes are not part of the web service model. These virtual nodes are Inputs and Outputs. The service's parameters are grouped under the virtual nodes Inputs and Outputs based on their types. Thus, this panel displays the inputs and outputs details of a selected service, e.g. PhotoAlbumService has an input: AlbumPhotos, and outputs: PhotoAlbum, PhotoAlbumSize, and PhotoAlbumColours.

The second panel also has a context sensitive property pane for each of the nodes, which allows users to add or remove parameters of a service. Two functions were implemented for this property pane, namely the add parameter (input or output), and the remove parameter. Depending on which node is selected, the appropriate action is enabled.


Data Flow

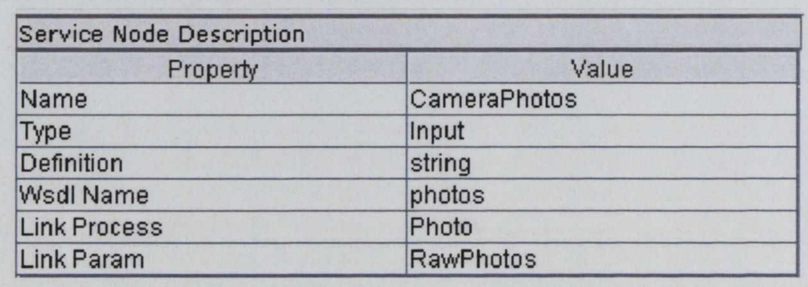
Process	Parameter	Process	Parameter
PhotoService	RawPhotos	PhotoAlbumPrintProcess	CameraPhotos
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	CameraPhotoCategory
PhotoService	PhotoSize	PhotoAlbumPrintProcess	CameraPhotoSize
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos
PrintService	DocumentName	PhotoAlbumService	PhotoAlbum
PrintService	NumberOfPages	PhotoAlbumService	PhotoAlbumSize
PrintService	NumberOfColours	PhotoAlbumService	PhotoAlbumColours

Figure 4-5, Web service data flow detail panel

The third panel, shown in Figure 4-5, displays the dataflow information for a composite service using a table with four columns. It displays the dataflow mapping between two parameters of two different services within a composite service or mapping between parameters of a constituent service and the composite service itself. This table displays the origin service in its first column, the origin service's parameter in the second column, the destination service in the third column, the destination service's parameter in the fourth and final column. For example, the last line informs the user that the NumberOfColours parameter of the Print Service maps to the PhotoAlbumColours parameter of the PhotoAlbum Service.

4.3.3 Web Service Editor Component

The web service editor component allows users to view and edit the web service semantic description. It provides a graphical user interface to aid developers in semantically describing web services. This component also uses Java Swing widgets and follows the MVC pattern. This component has a panel that is context sensitive, used for displaying and editing the details of a selected service node from the Web Service Viewer component. Hence, service nodes or their (input and output) parameters can be added to the panels in the Web Service Viewer component, and the details of these nodes can be edited here. This panel uses a table widget with two columns; where the first column shows the node's properties and the next column shows the values of these properties. Figure 4-6 shows the details of the input parameter called CameraPhotos. This is an input parameter for the PhotoAlbumPrint Service. The panel depicts the parameter type, its grounding information, and its dataflow information mapping to Photo Service.


Service Node Description

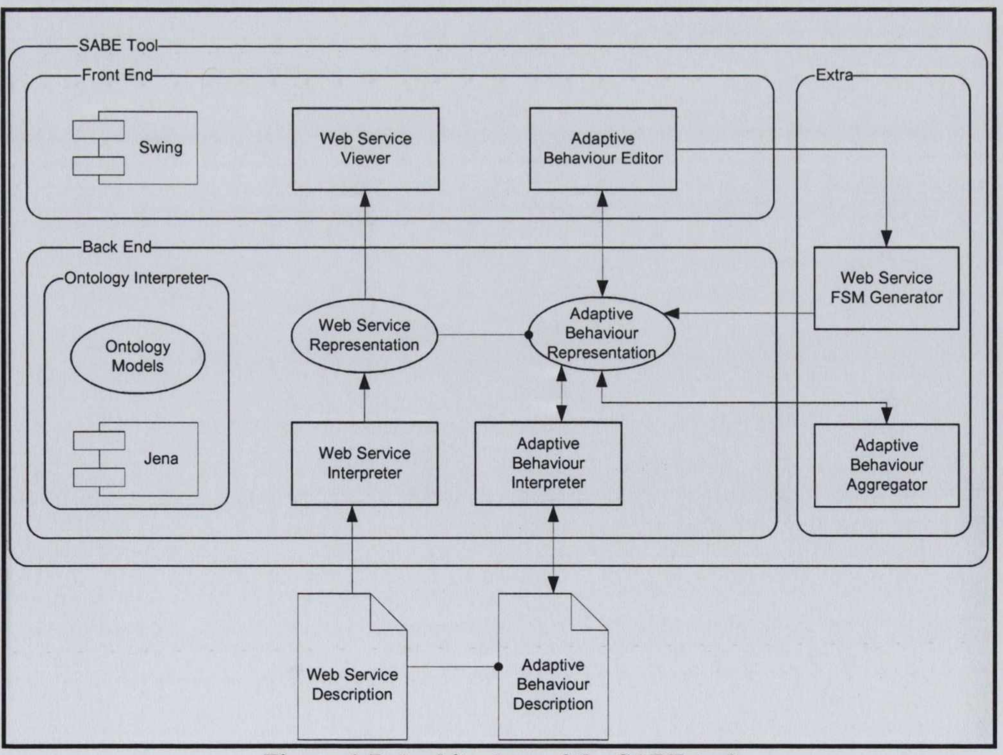
Property	Value
Name	CameraPhotos
Type	Input
Definition	string
Wsdl Name	photos
Link Process	Photo
Link Param	RawPhotos

Figure 4-6, Web service node description detail panel

4.4 Design and Implementation of the SABE tool

The Service Adaptive Behaviour Editor (SABE) tool allows web service developers to semantically describe the adaptive behaviours of adaptive web services. It consumes semantic web service descriptions (OWL-S) in order to reason about these web services. This tool allows developers to describe each of the adaptive behaviours pertaining to these web services and produces a formal representation for them expressed as FSM. It assumes that these behaviours can be described using events, states, and transitions even if not necessarily implemented as such.

In order to use a consistent format for describing adaptive web services, FSM was modelled as Ontology model. It also has the beneficial advantage that this tool can use the same reasoner to reason about the descriptions of web service, as well as their adaptive behaviours. The descriptions of these adaptive behaviours (FSM) are also associated with the semantic web service description (OWL-S). This association is through the addition of a property, called hasFSM, to *process* class in the OWL-S model. Thereby the semantic web service description is enriched with a description of their adaptive aspects.

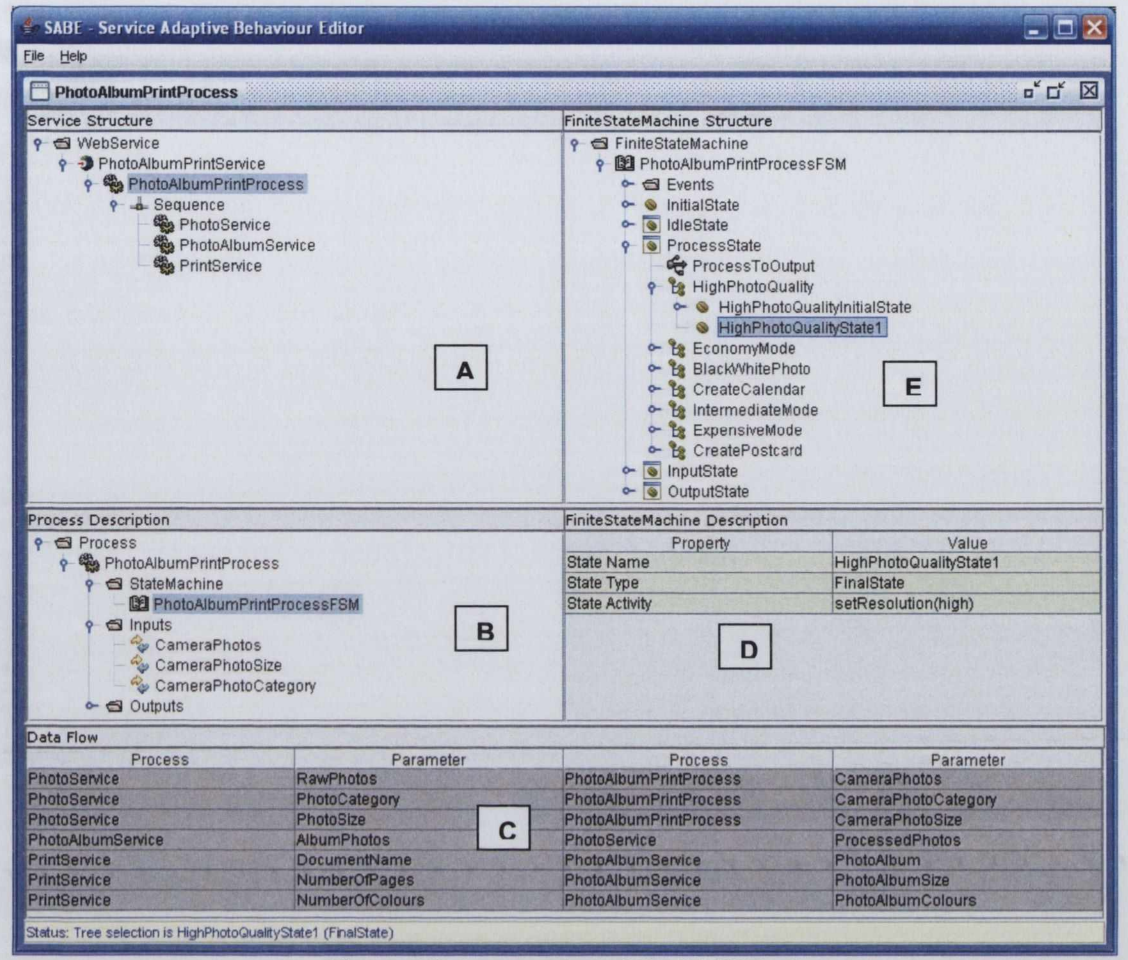

SABE Tool
Front End
Extra
Swing
Web Service Viewer
Adaptive Behaviour Editor
Back End
Ontology Interpreter
Web Service FSM Generator
Ontology Models
Web Service Representation
Adaptive Behaviour Representation
Jena
Web Service Interpreter
Adaptive Behaviour Interpreter
Adaptive Behaviour Aggregator
Web Service Description
Adaptive Behaviour Description
Figure 4-7, Architecture of the SABE tool

This tool was implemented as a component based architecture, as depicted in Figure 4-7, with the following main components:

- Ontology Interpreter Component, for interpreting and persisting adaptive web service's descriptions modelled using OWL;
- Web Service Viewer Component, for allowing users to view web service descriptions and their details;
- Adaptive Behaviour Editor Component, for allowing users to create and edit descriptions of adaptive behaviours belonging to web services;
- Web Service FSM Generator Component, for automatically generating a FSM model to contain sub-FSMs representing the web service's adaptive behaviours;
- Adaptive Behaviour Aggregator Component, for aggregating the FSMs representing adaptive behaviours of the constituent services for a composite service.

These components are described in detail in the sub-sections below.

The SABE tool was designed with a Graphical User Interface (GUI) that allows developers to view the particulars of web service's semantic descriptions (as OWL-S), as well as to browse and edit the details of FSM describing adaptive behaviours pertaining to the web service in question. Figure 4-8 shows a screenshot of the SABE tool.



Property	Value
State Name	HighPhotoQualityState1
State Type	FinalState
State Activity	setResolution(high)

Process	Parameter	Process	Parameter
PhotoService	RawPhotos	PhotoAlbumPrintProcess	CameraPhotos
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	CameraPhotoCategory
PhotoService	PhotoSize	PhotoAlbumPrintProcess	CameraPhotoSize
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos
PrintService	DocumentName	PhotoAlbumService	PhotoAlbum
PrintService	NumberOfPages	PhotoAlbumService	PhotoAlbumSize
PrintService	NumberOfColours	PhotoAlbumService	PhotoAlbumColours

Status: Tree selection is HighPhotoQualityState1 (FinalState)

Figure 4-8, A screenshot of the SABE tool

The following panels are depicted in Figure 4-8 (seen in an anti-clockwise rotation starting with the top left panel):

A. Web service composition detail panel, which displays the composition of composite web services i.e. control flow;

B. Web service process detail panel, which displays the details of a selected service (termed as process in OWL-S);

C. Web service data flow detail panel, which displays the data flow of a composite service;

D. FSM description detail panel, which allows users to edit the details of a selected node from the FSM, e.g. specifying the action of a state;

E. FSM structure detail panel, which allows users to add and edit FSMs to represent the service's adaptive behaviours.

4.4.1 Ontology Interpreter Component

The Ontology Interpreter component provides the functionality for parsing, interpreting, and saving semantic descriptions of adaptive web services based on OWL. These semantic descriptions are semantic web service description (OWL-S) and their adaptive behaviour's description (FSM). It uses Jena to parse, interpret, and save ontological descriptions related to adaptive web services. This is the same component used by WSDE tool, however enhanced for interpreting semantic descriptions of adaptive behaviours.

This component has been enhanced to have two sub-components to interpret each of the Ontology models used by the SABE tool: web service interpreter (see section 4.4.1), and adaptive behaviour interpreter. These sub-components contain the necessary Ontology schema to reason about the relevant descriptions. They also contain functions (logic code) to read from OWL description files, interpret OWL descriptions, and also to save semantic descriptions to OWL files. One sub-component was implemented to interpret the semantic web service descriptions as OWL-S. The other sub-component was implemented to interpret the ontology used to describe adaptive behaviours modelled as FSM.

To improve the visualisation of these descriptions and conform to MVC pattern, an internal representation for each of the models used was created. These internal representations are populated once their respective models are interpreted. In the same manner, information is extracted from them when persisting (saving) these descriptions; information which is translated into Ontology models according to their Ontology schema. This provides the flexibility to use an alternative model representation in the future if so desired.

A syntactical verification of the FSM model is performed during the persisting process to ensure that the FSM model of the adaptive behaviour described by the user is syntactically complete. This verification process checks that every state of the appropriate type has an action, that every transition has its details described, and that every event contains an event source. It also checks that every FSM has an initial state and that a transition is assigned to the relevant states, e.g. an initial state must have a single transition with this state as its source, and that no transition was assigned to a final state, i.e. a transition with a final state as its source state.

Once the FSM model passes the syntactical verification, this model is persisted in an OWL file and the sub-component returns to the user a report of success. In case the verification fails, it returns a report with a description of the fault in the FSM model. For example, the initial state of the HighPhotoQuality adaptive behaviour must have at least one transition, or State1 of this adaptive behaviour is missing its activity value.

4.4.2 Web Service Viewer Component

The Web Service Viewer component, described in section 4.4.2, displays the details of web services to the user using three panels. The first panel displays the control flow of a web service in case this service is a composite one. The second panel displays the details of a selected service. The third panel displays the dataflow of a service in case this service is a composite service. This component was enhanced to operate in read only mode for the SABE tool, which prevents users using the SABE tool from modifying the web service description.

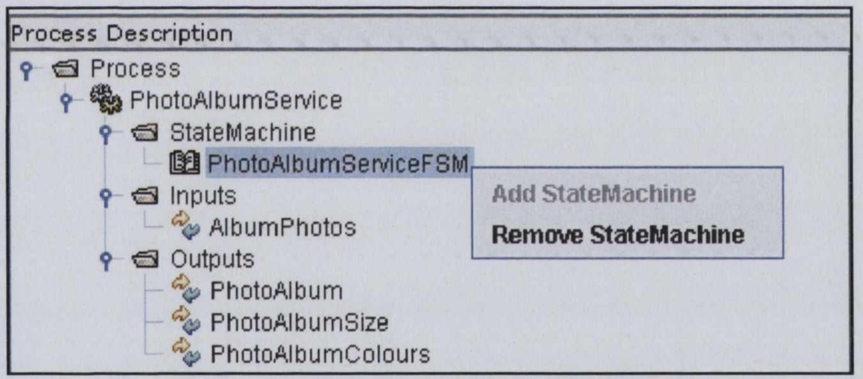


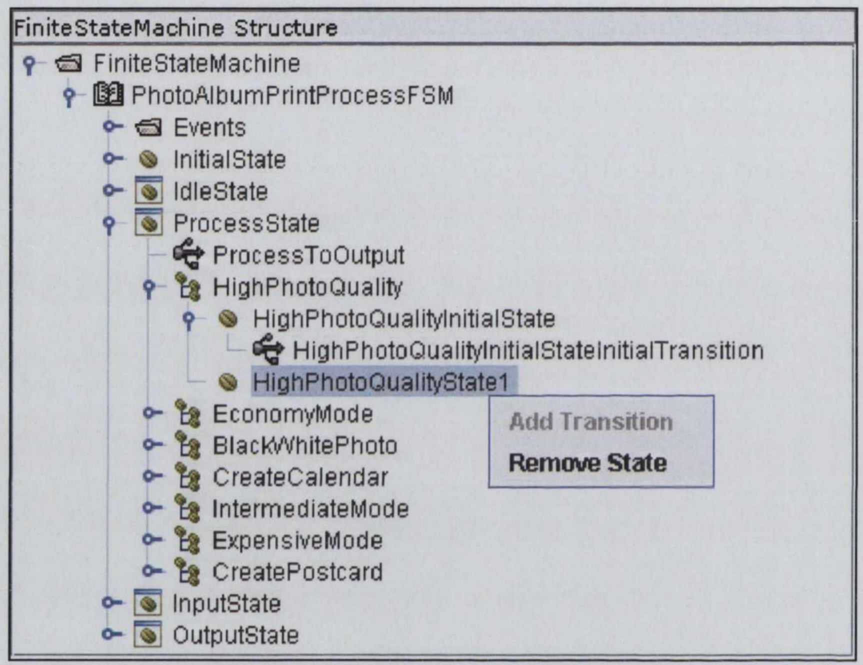
Figure 4-9, Web service process detail panel

In addition, the second panel, depicted in Figure 4-9, has been updated to display the details of adaptive web services. This panel, now has an extra virtual node called StateMachine. The StateMachine node can contain a single node representing the FSM model of this web service's adaptive behaviours, e.g. a node called PhotoAlbumServiceFSM.

This panel has a context sensitive property pane for the StateMachine node, which allows users to remove the FSM model created for this web service and to generate the FSM model to contain the descriptions of adaptive behaviours for this web service, and in cases of composite web services it automatically aggregates the FSM, describing the adaptive behaviours pertaining to their constituent web services.

4.4.3 Adaptive Behaviour Editor Component

This component allows users to view and edit adaptive behaviours belonging to adaptive web services modelled as FSM. It provides a graphical user interface to aid developers in describing adaptive behaviours of web services. This component also uses Java Swing widgets and follows the MVC pattern. This component has two panels; one panel for displaying and editing the structure of the FSM model using a tree widget; and a second panel for displaying and editing the different FSM model aspects using a table.

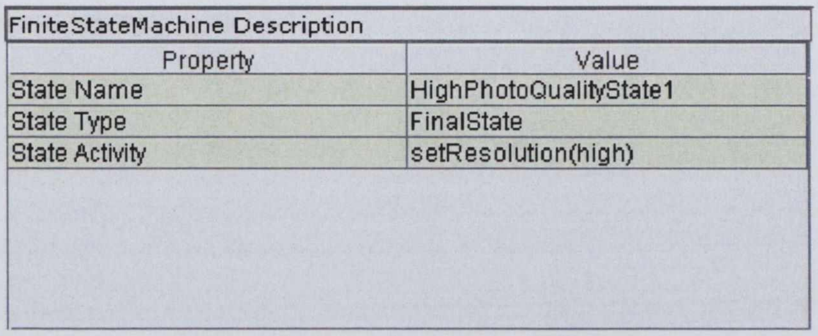


```
FiniteStateMachine Structure
FiniteStateMachine
  PhotoAlbumPrintProcessFSM
    Events
    InitialState
    IdleState
    ProcessState
      ProcessToOutput
      HighPhotoQuality
        HighPhotoQualityInitialState
          HighPhotoQualityInitialStateInitialTransition
        HighPhotoQualityState1
      EconomyMode
      BlackWhitePhoto
      CreateCalendar
      IntermediateMode
      ExpensiveMode
      CreatePostcard
    InputState
    OutputState
```

Figure 4-10, FSM structure detail panel

The first panel, depicted in Figure 4-10, displays the structure of the FSM model describing the adaptive behaviours of web services using a tree widget. The FSM model generated for a web service contains four composite states, namely IdleState, InputState, ProcessState, and OutputState. Sub-FSMs representing adaptive behaviours can be added under these states. For example, this panel is displaying the adaptive behaviours under ProcessState, such as HighPhotoQuality and EconomyMode among others. This panel has a context sensitive property pane which changes depending on the node selected and the child node it contains. The main functions provided by this property pane are to create valid child node types or to delete the node itself.

This panel aids users in creating a FSM for describing adaptive behaviours. More importantly, it attempts to prevent users from creating a functionally incorrect model. For example, a property pane for HighPhotoQualityState1 provides only the option to remove the state since this is a state of type FinalState, and final states should not have any transitions originating from them, i.e. transitions with these states as their source state, therefore this component prevents users from adding a transition to such state.



FiniteStateMachine Description

Property	Value
State Name	HighPhotoQualityState1
State Type	FinalState
State Activity	setResolution(high)

Figure 4-11, FSM description detail panel

The second panel is context sensitive and displays the details of a selected node (e.g. HighPhotoQualityState1) from the first panel. This panel uses a table widget with two columns, where the first column shows the node's properties and the next column shows the values of these properties. Figure 4-11 shows this panel in action for a selected node where its type is FinalState and this state activity is "setResolution(high)". This panel allows users to edit the properties of a selected node and aids the user when editing by providing a dropdown menu with appropriate options, for example, a dropdown menu of the different types of states. It also highlights in red properties that are mandatory and not yet completed.

4.4.4 Web Service FSM Generator Component

The Web Service FSM Generator component provides the functionality for generating a FSM model for a web service so as to contain sub-FSMs representing adaptive behaviours pertaining to this web service. This component produces a FSM with four composite states, namely IdleState, InputState, ProcessState, and OutputState, representing the various states of a running web service [31]. Sub-FSMs can then be added to these states, by the user, to represent the different adaptive behaviours of a web service, arranged according to the states provided. This component generates transitions that link these states together, and events that trigger these transitions. Figure 4-12 shows a diagram of this FSM model. Adaptive behaviours shown in Figure 4-10, such as EconomyMode, would be attached to the Process State node in this diagram.

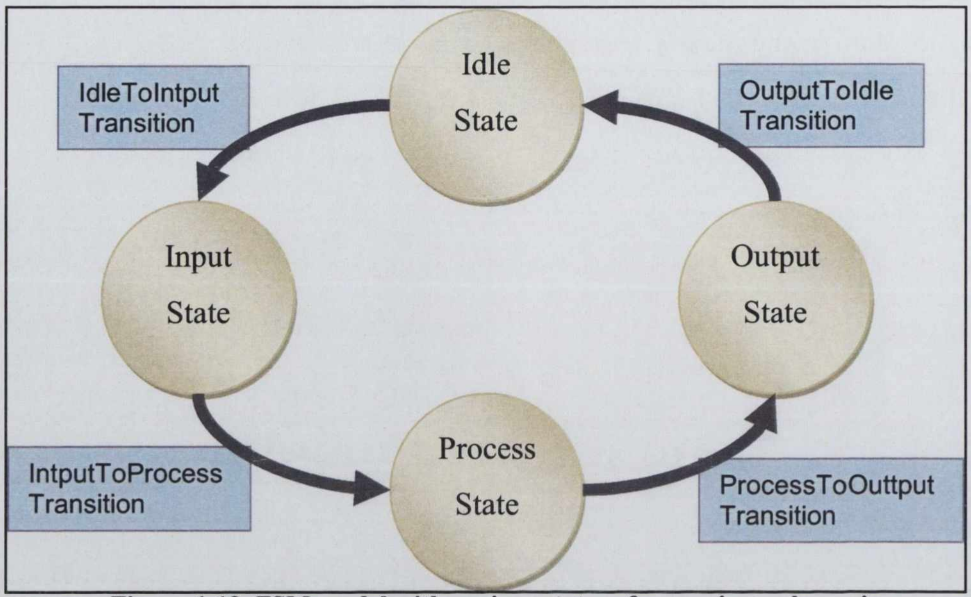


Figure 4-12, FSM model with various states of a running web service

4.4.5 Adaptive Behaviour Aggregator Component

The Adaptive Behaviour Aggregator component provides the functionality for automatically aggregating the adaptive behaviours for adaptive composite services. It automatically aggregates the FSM models representing the adaptive behaviours belonging to the constituent adaptive web services for a composite web service. This component utilises the internal representation of web services and the internal representation of their adaptive behaviours to automatically aggregate these adaptive behaviours for composite services. These adaptive behaviours are aggregated in a reversed hierarchical manner, i.e. they are composed together at each level of the hierarchy, starting from the leaves and working to the parent node(s) of the service composition hierarchy.

The algorithm for this automatic aggregation of adaptive behaviours is as follows:

1. Search for atomic web services within the composite web service's description starting from the topmost web service;
2. Once all the atomic services are found, retrace back to the topmost web service;
3. On the return path, if any web service has a FSM associated to it
    a. Generate a FSM for the parent web service (if one is not present)

b. Search each of the states (Idle, Input, Process, Output) for sub-FSMs, describing adaptive behaviours of this web service;

c. For every FSM found, copy all its contents (state, event, transitions) to the parent web service;

This algorithm aggregates the FSMs representing the adaptive behaviours belonging to the constituent atomic services for a composite web service. The composite web service's semantic description is then associated with a resultant FSM containing all the adaptive behaviours pertaining to its constituent adaptive atomic services.

4.5 SMPE tool and its components

The Service Management Policy Editor (SMPE) tool allows web service managers or web service users to specify management policies, obligation type policies, which will manage adaptive web services. This tool consumes the semantic description of web services, and the formal representation of their adaptive behaviours. These inputs provide the vocabulary needed for specifying high-level management policies, and for the automated refinement process of these policies, generating low-level mapping and enforceable policies to manage adaptive web services.

The SMPE tool parses semantic web service descriptions of adaptive web services, described in OWL-S, and their adaptive behaviours descriptions, described as FSM. This tool was also implemented with a GUI, which allows its users to view details of web services, details of their adaptive behaviours, and more importantly to view and edit management policies for managing these adaptive web services. Once management policies are specified, this tool allows these policies to be automatically refined and the generated low-level policies are saved as Obligation Ontology model to an OWL file. This tool also persists (saves) the management policies specified to a file so that users can load those policies into the tool in the future.

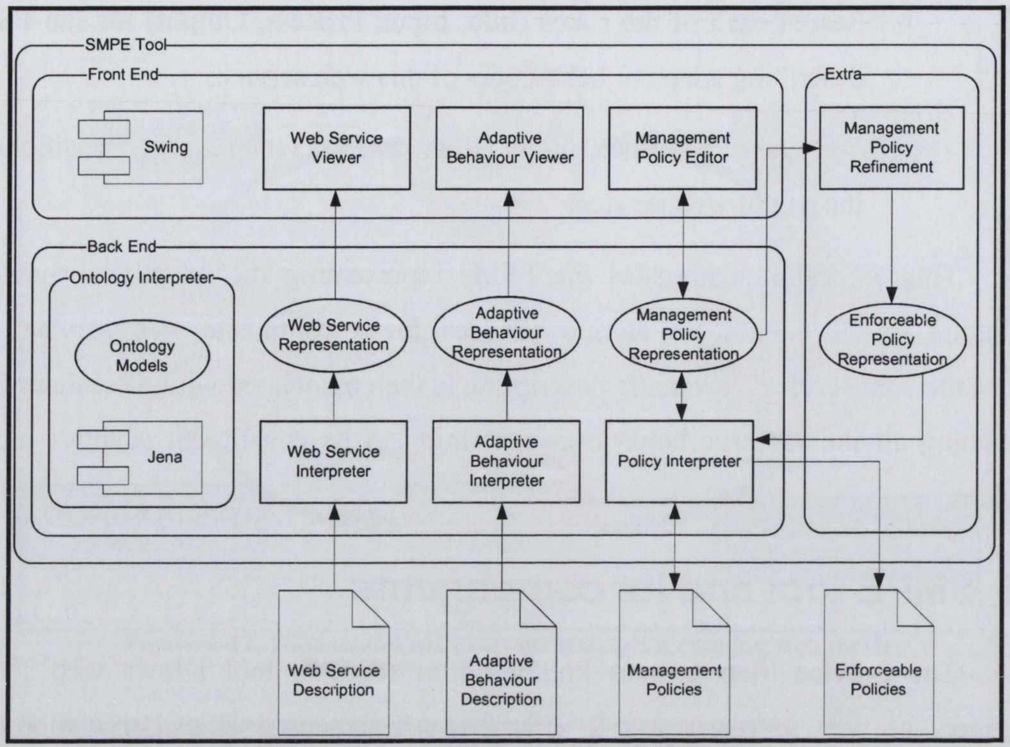

SMPE Tool
Front End
Swing
Web Service Viewer
Adaptive Behaviour Viewer
Management Policy Editor
Extra
Management Policy Refinement
Back End
Ontology Interpreter
Ontology Models
Jena
Web Service Representation
Adaptive Behaviour Representation
Management Policy Representation
Enforceable Policy Representation
Web Service Interpreter
Adaptive Behaviour Interpreter
Policy Interpreter
Web Service Description
Adaptive Behaviour Description
Management Policies
Enforceable Policies

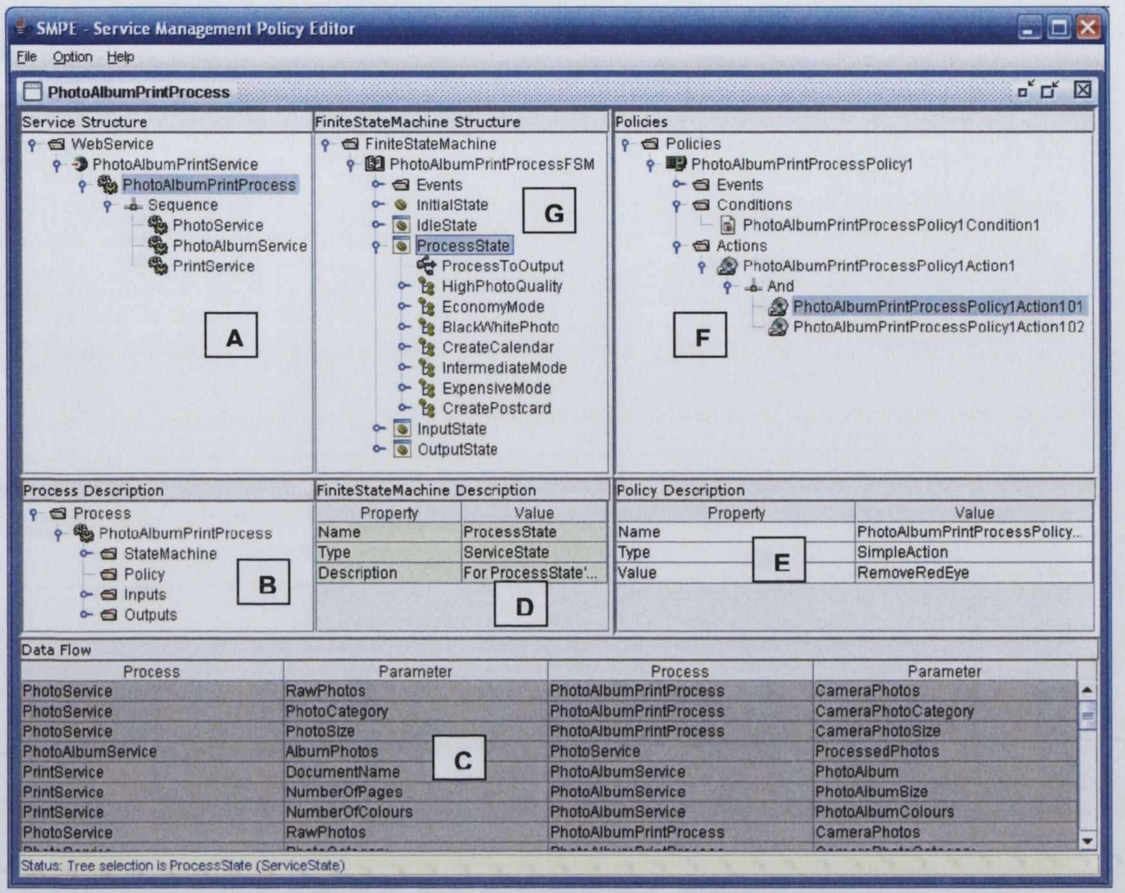
Figure 4-13, Architecture of the SMPE tool

This tool was implemented on component based architecture, as depicted in Figure 4-13, with the following main components:

- Ontology Interpreter Component, for interpreting and persisting adaptive web service's descriptions and management policies modelled using OWL;
- Web Service Viewer Component, for allowing users to view the details of web service semantic descriptions;
- Adaptive Behaviour Viewer Component, for allowing users to view descriptions of adaptive behaviours pertaining to adaptive web services;
- Management Policy Editor Component, for allowing users to specify high-level management policies for adaptive web services;
- Management Policy Refinement Component, for automatically refining high-level management policies, and generating low-level enforceable policies as well as mapping policies which maps them together.

These components are further described in the sub-sections below.

This tool was designed with a GUI to assist users in specifying the particulars of high-level management policies for managing the adaptive behaviours of adaptive web services. Figure 4-14 shows a screenshot of the SMPE tool.



```
Service Structure
WebService
  PhotoAlbumPrintService
    PhotoAlbumPrintProcess
      Sequence
        PhotoService
        PhotoAlbumService
        PrintService
```

```
FiniteStateMachine Structure
FiniteStateMachine
  PhotoAlbumPrintProcessFSM
    Events
    InitialState
    IdleState
    ProcessState
      ProcessToOutput
      HighPhotoQuality
      EconomyMode
      BlackWhitePhoto
      CreateCalendar
      IntermediateMode
      ExpensiveMode
      CreatePostcard
    InputState
    OutputState
```

```
Policies
Policies
  PhotoAlbumPrintProcessPolicy1
    Events
    Conditions
      PhotoAlbumPrintProcessPolicy1Condition1
    Actions
      PhotoAlbumPrintProcessPolicy1Action1
        And
          PhotoAlbumPrintProcessPolicy1Action101
          PhotoAlbumPrintProcessPolicy1Action102
```

```
Process Description
Process
  PhotoAlbumPrintProcess
    StateMachine
    Policy
    Inputs
    Outputs
```

FiniteStateMachine Description

Property	Value
Name	ProcessState
Type	ServiceState
Description	For ProcessState'...

Policy Description

Property	Value
Name	PhotoAlbumPrintProcessPolicy...
Type	SimpleAction
Value	RemoveRedEye

Data Flow

Process	Parameter	Process	Parameter
PhotoService	RawPhotos	PhotoAlbumPrintProcess	CameraPhotos
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	CameraPhotoCategory
PhotoService	PhotoSize	PhotoAlbumPrintProcess	CameraPhotoSize
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos
PrintService	DocumentName	PhotoAlbumService	PhotoAlbum
PrintService	NumberOfPages	PhotoAlbumService	PhotoAlbumSize
PrintService	NumberOfColours	PhotoAlbumService	PhotoAlbumColours
PhotoService	RawPhotos	PhotoAlbumPrintProcess	CameraPhotos

Status: Tree selection is ProcessState (ServiceState)

Figure 4-14, A screenshot of the SMPE tool

The following panels are depicted in Figure 4-14 (seen in an anti-clockwise rotation starting with the top left panel):

A. Web service composition detail panel, which displays the composition of composite web services i.e. control flow;

B. Web service process detail panel, which displays the details of a selected service (termed as process in OWL-S);

C. Web service data flow detail panel, which displays the data flow of composite services;

D. FSM description detail panel, which displays details of a selected node from the FSM, e.g. the action of a state node;

E. Policy aspect description panel, which allows users to edit the selected aspect of a management policy;

F. Policy Structure detail panel, which allows users to specify their management policies;

G. FSM structure detail panel, which displays service's adaptive behaviours as FSM

4.5.1 Ontology Interpreter Component

The Ontology Interpreter component provides the functionality for parsing, interpreting, and saving semantic descriptions based on OWL. This component is an extended version of that used in the SABE tool. It reads semantic descriptions of web services and their adaptive behaviours as before. However, this component is restricted from saving these descriptions when used in this tool. Instead, it has been extended to read and save instances of an Obligation Policy Ontology model to an OWL file in lexical form.

The component was originally implemented with two sub-components to interpret the semantic web service descriptions as OWL-S, and to interpret the ontology used to describe adaptive behaviours modelled as FSM (see section 4.4.1). For the SMPE tool, this component has been extended with a new sub-component, which interprets the ontology model used to describe management policies, mapping policies and enforceable policies.

Once these ontology policy models are interpreted, their respective internal representations are populated. And when saving these policies, their details are extracted from these internal representations and translated into Ontology models according to their Ontology schema. A syntactical verification of the policy description is performed during the persisting process to make sure that the management policy specified and the enforceable policies generated are syntactically complete. This verification process checks that every aspect of these policies is described.

4.5.2 Web Service Viewer Component

The web service viewer component uses Java Swing widget panels to display the details of the web service's semantic description. This component is the same component used in SABE tool; see section 4.4.2 for its description. In summary, this component displays the details of web services to the user using three panels. The first panel displays the control flow of a composite service. The second panel is a context sensitive panel which displays the details of a service selected from the first panel. Lastly, the third panel displays the dataflow of a composite service.

However, this component's second panel operates in read-only mode. The functions for creating and deleting FSM for web services are disabled. Thus, it prevents managers and non-developers from incorrectly adding or removing FSM using this tool.

4.5.3 Adaptive Behaviour Viewer Component

The Adaptive Behaviour Viewer component provides a graphical user interface to aid service managers in browsing the adaptive behaviours of web services expressed as FSM. It uses Java Swing widgets and follows the MVC pattern. Two panels display the FSM describing adaptive behaviours belonging to a web service. The first panel displays the structure of the FSM model, while the second panel is a sensitive context panel that displays the different aspects of the FSM model. This component is the Adaptive Behaviour Editor component, used in the SABE tool (see section 4.4.3), operating in read-only mode.

4.5.4 Management Policy Editor Component

The Management Policy Editor component provides a graphical user interface to support service managers in specifying management policies to manage the adaptive behaviours of adaptive (composite) services. This component uses Java Swing widgets and follows the MVC pattern. Two panels allow users to view and specify management policies for adaptive web services. The first panel is for displaying and editing the different aspects of management policies using a tree widget. The second panel is for displaying and editing the details of each of the aspects of a management policy using a table widget. This component only allows management policies to be specified to the topmost (atomic or composite) services, thus preventing managers from inappropriately specifying policies directly to constituent web services.

Note that the SMPE tool allows users to specify policies for single atomic services. But, the SMPE does not allow the specification of policies to atomic services when these atomic services are being used by some composite service. The tool ensures that policies must always be specified for the topmost service. However, users can be assured that if they want to change the behaviour of a particular constituent atomic service, they just need to specify the appropriate policy to the composite service and it will percolate down to the relevant constituent service during the refinement process.

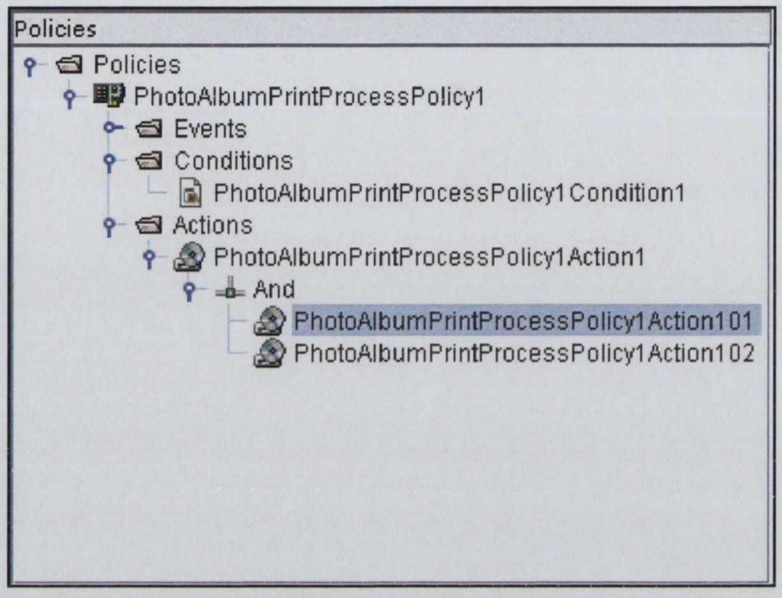
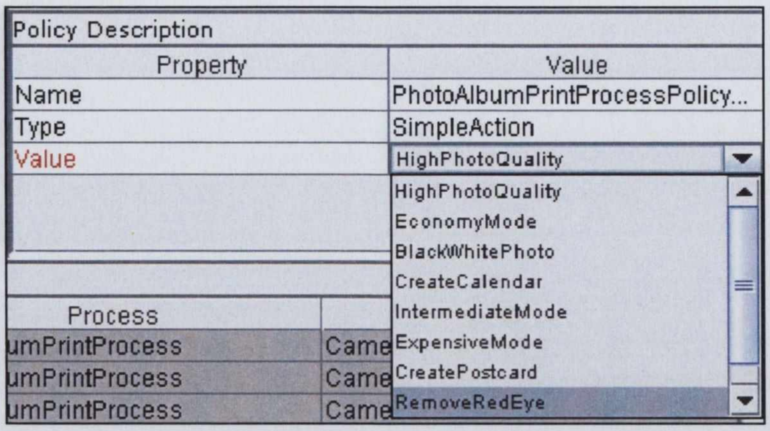

Policies
Policies
PhotoAlbumPrintProcessPolicy1
Events
Conditions
PhotoAlbumPrintProcessPolicy1Condition1
Actions
PhotoAlbumPrintProcessPolicy1Action1
And
PhotoAlbumPrintProcessPolicy1Action101
PhotoAlbumPrintProcessPolicy1Action102

Figure 4-15, Policy structure detail panel

In the first panel, depicted in Figure 4-15, policy descriptions are displayed with the use of a tree widget with customised icons for the different types of nodes. These policy nodes can be event nodes, condition nodes, or action nodes. This is due to the fact that the management policies supported by this tool are obligation type policies. These aspects are grouped, accordingly, under virtual nodes: events, condition, and actions. A node representing an action, condition, or event can be of type simple or complex; where complex nodes are those which can have other nodes joined by a Boolean node. Boolean nodes can have up to two child nodes and can only be of type AND or OR, allowing the policy's complexity to be expressed in a scalable manner. This panel has a context sensitive property pane which adapts depending on the node selected and the child node it contains. The functions most commonly used for this property pane are to create valid type child node or to delete the node itself. An exclusive function for invoking the policy refinement process on the management policies is also available.


Policy Description

Property	Value
Name	PhotoAlbumPrintProcessPolicy...
Type	SimpleAction
Value	HighPhotoQuality

Process	
umPrintProcess	Came
umPrintProcess	Came
umPrintProcess	Came

Figure 4-16, Policy aspect description panel

The second panel (shown in Figure 4-16) displays the details of the node selected in the first panel representing one of the actions of a management policy. This panel uses a table widget with two columns, where the first column shows the properties and the next column shows the values. This panel allows users to edit the details of a selected node, and highlights in red properties that are mandatory and incomplete. For example, the value property for this action node (depicted in Figure 4-16) is highlighted in red since a choice has not been made from the dropdown list of possible high-level actions (adaptive behaviours). Users are aided when editing by the provision of a dropdown list with appropriate options. In fact, it assures that the management policies specified for adaptive web services use only the valid vocabulary acquired from their semantic descriptions. Thus, the action aspect of management policies is restricted to the adaptive behaviours described for adaptive web services. The condition aspect of management policies is restricted to the semantic description of web service's parameters, i.e. inputs and outputs.

4.5.5 Management Policy Refinement Component

The management policy refinement component provides the functionality for automatically refining specified management policies and automatically generating low-level policies to manage adaptive web services. This component utilises the web service descriptions and the formal representation of their adaptive behaviours to achieve this objective. Each management policy is refined by analysing its aspects (event, condition, and action) and generating mapping policies for the immediate constituent services, which are then further refined in a hierarchical manner until enforceable policies are created for the relevant constituent atomic services.

This component has two modes of policy refinement, action-only refinement which only refines the action aspects of management policies, and condition-action refinement which refines the condition, as well as the action aspects of management policies. Users can choose a policy refinement mode through the SMPE tool's menu before performing the refinement process. Both policy refinement modes use events as a means to link the policies together.

Automatic action-only policy refinement algorithm

The action-only refinement mode assumes that the condition aspects of the management policies can be handled by the topmost composite service itself. Thus, it refines only the action aspects of management policies. The algorithm for the automatic refinement of the action aspects of management policies has two parts. The first part generates mapping policies in a hierarchical manner for composite web services. This algorithm is as follows:

1. Copy the contents of a management policy into a new mapping policy since the mapping policy will be modified during the refinement process, a copy of the original needs to be retained for the user;
2. Search for the adaptive behaviour named as the action of this mapping policy in the FSM of the immediate constituent web services;
3. If the named adaptive behaviour is found, generate a new mapping policy for this web service, naming this adaptive behaviour as its action and the relevant service as its target;
4. Repeat for all the actions in the original policy;
5. Generate an event e.g. E1 and insert it into the new mapping policies;
6. Replace the action of the original mapping policy with a new action that generates this new event, e.g. sendEvent(E1);
7. Repeat step 2 for all the new mapping policies.

This algorithm generates mapping policies which target the relevant constituent atomic web services and all web services in the hierarchical path between them and the topmost web service i.e. the composite web service itself.

The second part of this algorithm generates the enforceable policies for the relevant constituent atomic web services, and starts by searching for mapping policies targeting atomic web services and for each policy found:

1. Search for the FSM description of the adaptive behaviour named as the action of this mapping policy;
2. Find the initial state in this FSM;
3. Generate new enforceable policies for every transition deriving from this state;
4. Populate the event aspect of these policies with new generated event and event described in the transition, recording the generated event for the first enforceable policy, e.g. E12;
5. Populate the condition aspect of these enforceable policies with the guard description of their relevant transitions;
6. Populate the action aspect of these enforceable policies with the activity described in the target state of the relevant transition;
7. In case the target state of a transition is not a final type state, add an extra action to the derived policy that triggers the event used by the next policy in the chain, i.e. enforceable policy derived from the transition whose state source is this particular state.
8. Follow the state machine process path of this FSM, identifying the next state and repeating step 4;
9. Replace the action of the original mapping policy with a new action that generates the recorded event, e.g. sendEvent(E12);

This algorithm systematically creates a policy for every transition in the FSM describing an adaptive behaviour. Enforceable policies are generated, each with an event which triggers it, a condition based on the transition's guard, which must be satisfied, and an action based on the described activity of the transition's target state, which will be performed.

Automatic condition-action policy refinement algorithm

The condition-action refinement mode assumes that the topmost composite service is a virtual service and that the condition aspect of the management policies needs to be handled by the relevant constituent services. Thus, it also refines the condition aspects of management policies.

The automatic refinement of condition aspects of management policies follows the same principle as the refinement process of action aspects, albeit on a far more complex level. This algorithm has four parts, where parts three and four are the same as the action-only policy refinement algorithm. The first part generates mapping policies for the condition aspects of management policies in a hierarchical manner for composite web services and is as follows:

1. Copy the contents of a management policy into a new mapping policy;
2. Search the dataflow description of this web service for the mapping details of the web service's parameters used as the subject of the condition aspect of a policy;
3. Retrieve the equivalent parameter and its web service from the dataflow description, where this web service is a constituent web service;
4. Generate a mapping policy for the constituent web service found, and populate its condition aspect with this web service's parameter;
5. Generate an event, e.g. E1, and use it in the new mapping policy as its triggering event;
6. Remove the condition aspect of the original mapping policy and insert a triggering action of the generated event into this policy, e.g. sendEvent(E1);
7. Repeat step 2 for all the new mapping policies targeting the constituent web services;
8. Add a triggering action to the last mapping policy generated, which triggers the event of a new mapping policy assigned to parent web service, e.g. sendEvent(E12);
9. Add a new mapping policy to the parent web service with such an event, e.g. E12;
10. Repeat step 8 until it reaches the topmost web service;

This algorithm generates mapping policies that trigger enforceable policies, for monitoring the desired conditions on constituent web services. In addition, it generates mapping policies which triggers the top level policies if the relevant conditions are met.

The second part of this algorithm generates the enforceable policies for the relevant constituent atomic web services to monitor the desired conditions and is as follows:

1. Search for mapping policies targeting atomic web services;
2. Identify the service's parameters used as subject in the condition aspect of these mapping policies;
3. Search the web service's description for the mapping of these parameters to their WSDL counterparts;
4. Convert these mapping policies into enforceable policies, changing their condition aspects to use the WSDL counterpart of the web service's parameter being used;

This algorithm generates enforceable policies to monitor parameters of atomic services, grounded as WSDL parameters. These policies monitors for the desired conditions specified in the management policy. This algorithm then uses the action-only refinement algorithm to refine the action aspect of management policies, i.e. parts 3 and 4 of this algorithm.

4.6 A Walkthrough in Using SABE to Describe Adaptive Web Services

The SABE tool was designed and implemented to allow developers to formally describe the adaptive behaviours of web services. So far, a lengthy description of the design and implementation of this tool was presented, which can be seen as complex and convoluted. However, the SABE tool was designed with a GUI that shields users from the semantic description's lexical form, and automated features to facilitate its users. This section demonstrates how this tool can be used by a service developer to describe the adaptive behaviours of an atomic and a composite service. The SABE tool was designed with a tree and a table widget which is familiar to most computer users, and it is evaluated by a usability evaluation in the next chapter (section 5.3).

This section provides a work example of how users can use the SABE to describe adaptive web services. It starts with a demonstration of how this tool is used to describe an adaptive behaviour for an atomic web service. Next, a demonstration of how this tool is used to aggregate the adaptive behaviours of a composite web service is presented. It provides illustrated example of how the SABE tool supports the *Web Service Adaptivity Description Activity* and *Web Service Adaptivity Aggregation Activity* of the MAWS methodology.

4.6.1 Describing Adaptive Behaviours of Atomic Web Services

The PrintService example, described earlier in section 3.3, is used to illustrate how the SABE tool is used to describe the EconomyMode adaptive behaviour of this service. EconomyMode is an adaptive behaviour that modifies the PrintService to print on both sided with two pages on each side i.e. four pages per sheet. It is assumed that this adaptive behaviour has been implemented. It also assumed that files containing OWL-S description of the web service is also available. The steps for describing this adaptive behaviour are as follows:

i. Start the SABE tool;
ii. Load the semantic description of the PrintService from PrintService.owl file;
iii. Add a FSM to represent the adaptive behaviours for this service;
iv. Add a FSM to describe EconomyMode adaptive behaviour;
v. Add the initial state for the EconomyMode FSM;
vi. Add the transitions in the EconomyMode FSM;
vii. Add the necessary states in the EconomyMode FSM;
viii. Save the FSM representing the adaptive behaviours of the PrintService;

Screenshots are used to visually aid in the sequence of actions performed on the SABE tool to describe such adaptive behaviour. Note that descriptions are above the screenshots.

i. Start the tool and select open (see Figure 4-17).

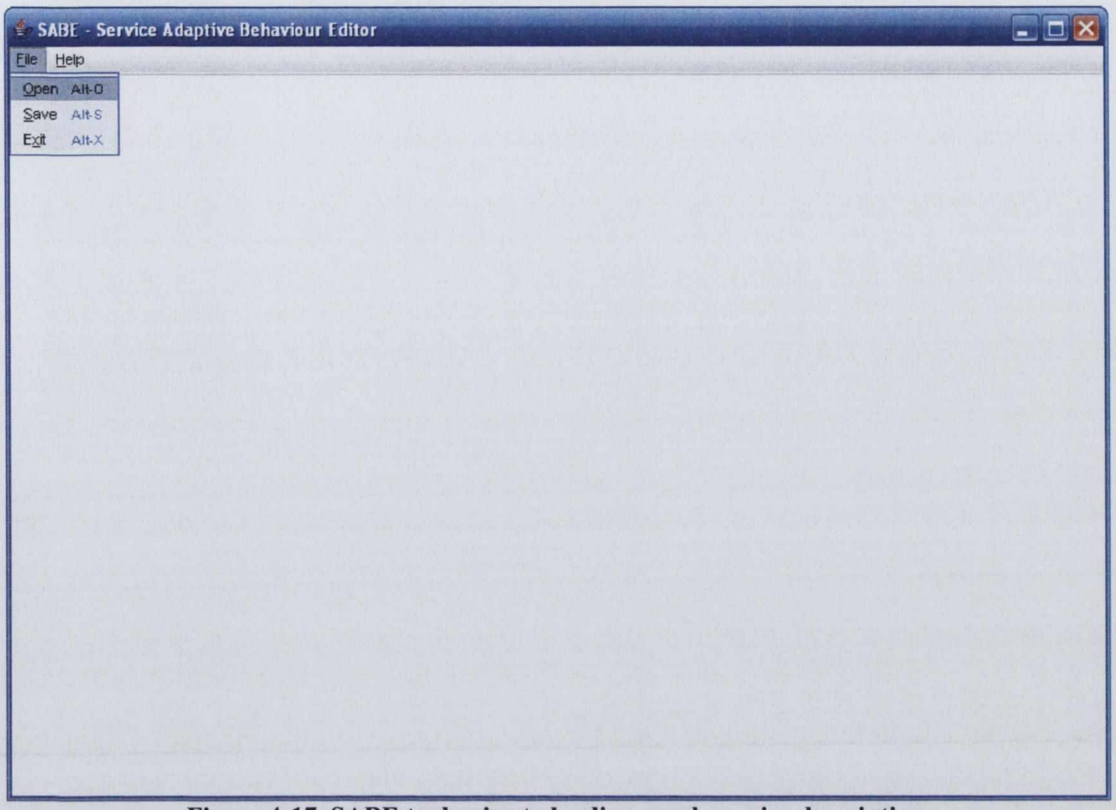


Figure 4-17, SABE tool prior to loading a web service description

ii. Select the PrintService.owl file (see Figure 4-18).

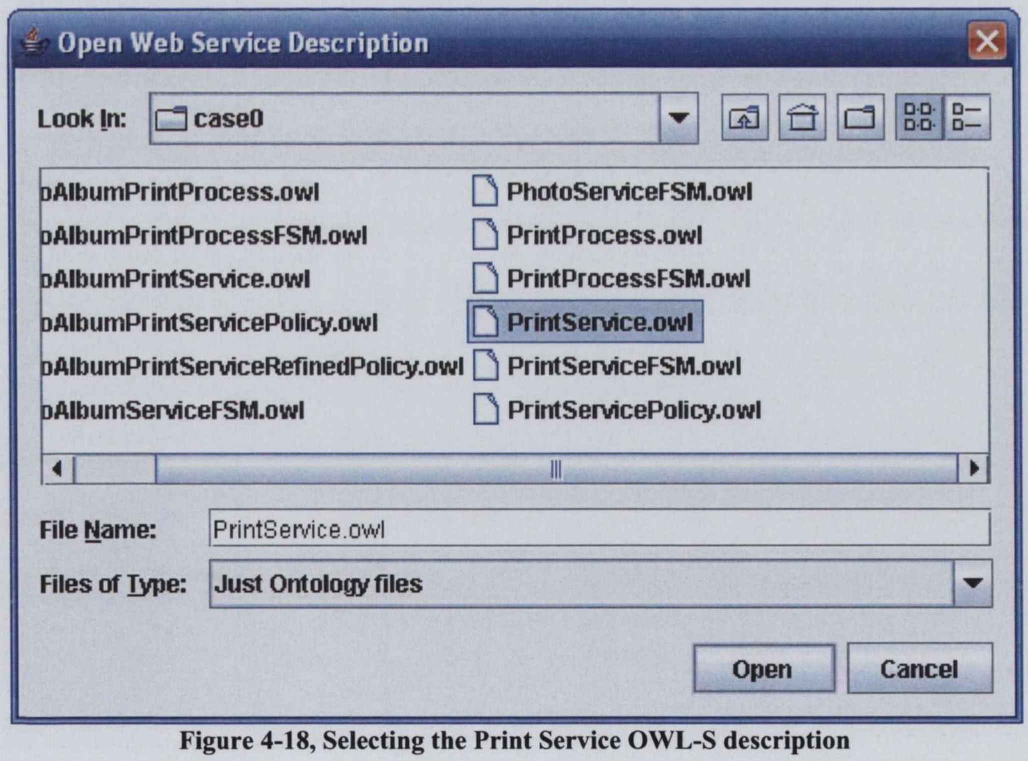
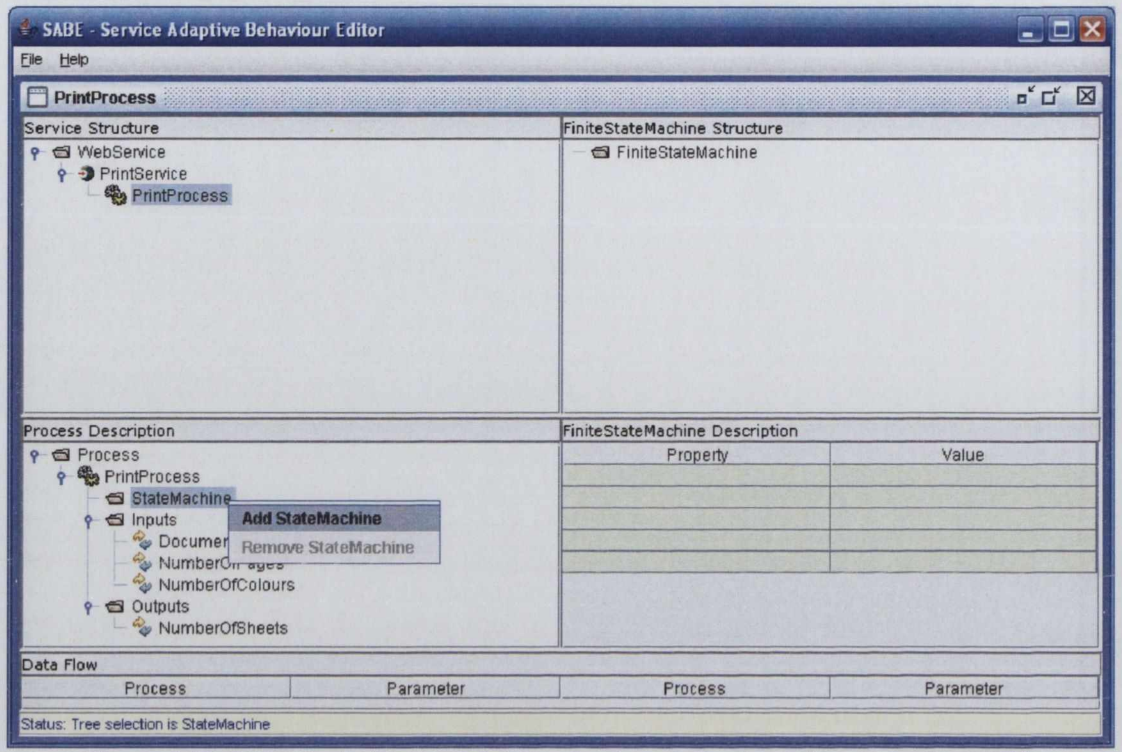


Figure 4-18, Selecting the Print Service OWL-S description

iii. Select PrintProcess on the top left panel, this will load the details of the service's process in the bottom left panel. Expand PrintProcess and right click on StateMachine to see a dropdown menu. Select Add StateMachine to add a FSM for this service (see Figure 4-19).



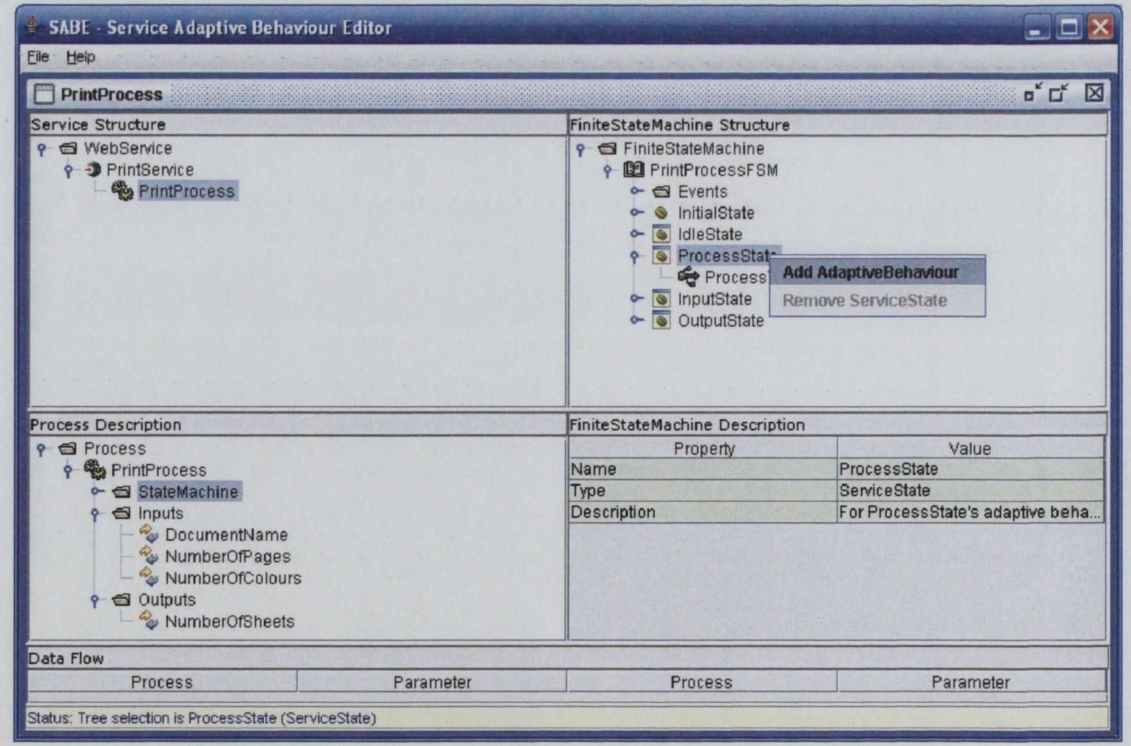
Property	Value

Process	Parameter	Process	Parameter

Status: Tree selection is StateMachine

Figure 4-19, SABE tool with the Print Service description loaded and creating FSM

iv. Expand the FSM created on the top right panel. Select ProcessState and right click to get a dropdown menu with an option to add an adaptive behaviour. Select the *Add AdaptiveBehaviour* option (see Figure 4-20).



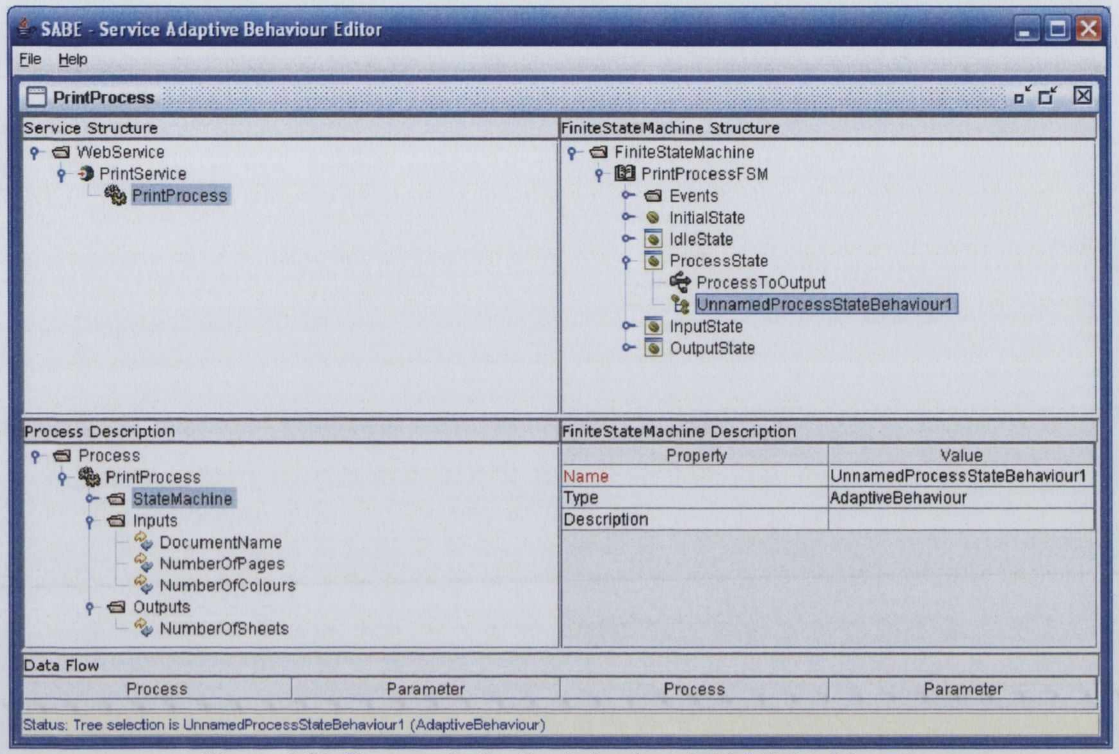
Property	Value
Name	ProcessState
Type	ServiceState
Description	For ProcessState's adaptive beha...

Process	Parameter	Process	Parameter

Status: Tree selection is ProcessState (ServiceState)

Figure 4-20, Creating a sub-FSM to describe an adaptive behaviour

Select this new node and see its properties load in the bottom right panel where required fields are highlighted in red (see Figure 4-21). Name it EconomyMode and add a description.



SABE - Service Adaptive Behaviour Editor

File Help

PrintProcess

Service Structure

- WebService
  - PrintService
    - PrintProcess

FiniteStateMachine Structure

- FiniteStateMachine
  - PrintProcessFSM
    - Events
    - InitialState
    - IdleState
    - ProcessState
      - ProcessToOutput
      - UnnamedProcessStateBehaviour1
    - InputState
    - OutputState

Process Description

- Process
  - PrintProcess
    - StateMachine
    - Inputs
      - DocumentName
      - NumberOfPages
      - NumberOfColours
    - Outputs
      - NumberOfSheets

FiniteStateMachine Description

Property	Value
Name	UnnamedProcessStateBehaviour1
Type	AdaptiveBehaviour
Description	

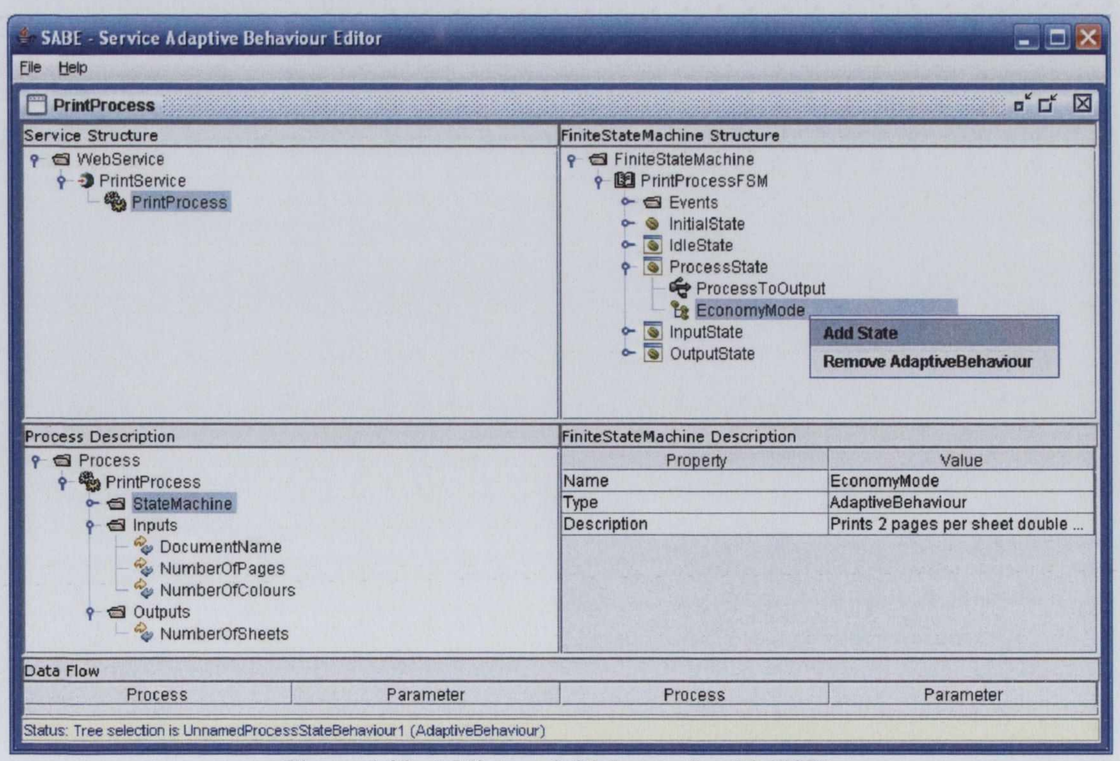
Data Flow

Process	Parameter	Process	Parameter

Status: Tree selection is UnnamedProcessStateBehaviour1 (AdaptiveBehaviour)

Figure 4-21, Unnamed adaptive behaviour is highlighted in red

v. Right click on EconomyMode node to add initial state to this adaptive behaviour (see Figure 4-22).



SABE - Service Adaptive Behaviour Editor

File Help

PrintProcess

Service Structure

- WebService
  - PrintService
    - PrintProcess

FiniteStateMachine Structure

- FiniteStateMachine
  - PrintProcessFSM
    - Events
    - InitialState
    - IdleState
    - ProcessState
      - ProcessToOutput
      - EconomyMode
    - InputState
    - OutputState

Process Description

- Process
  - PrintProcess
    - StateMachine
    - Inputs
      - DocumentName
      - NumberOfPages
      - NumberOfColours
    - Outputs
      - NumberOfSheets

FiniteStateMachine Description

Property	Value
Name	EconomyMode
Type	AdaptiveBehaviour
Description	Prints 2 pages per sheet double ...

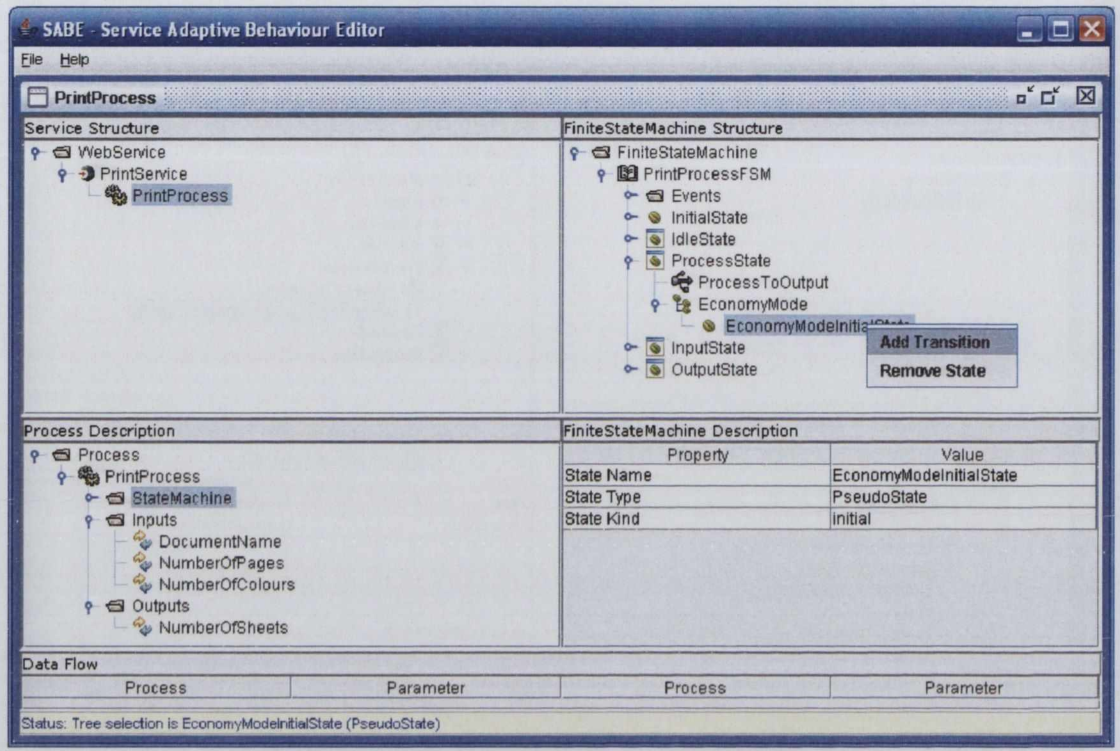
Data Flow

Process	Parameter	Process	Parameter

Status: Tree selection is UnnamedProcessStateBehaviour1 (AdaptiveBehaviour)

Figure 4-22, Adding an initial state for this FSM

vi. The first state for this adaptive behaviour (FSM) is an initial state which must have a transition. Select *Add Transition* from its dropdown menu to create a transition (see Figure 4-23).

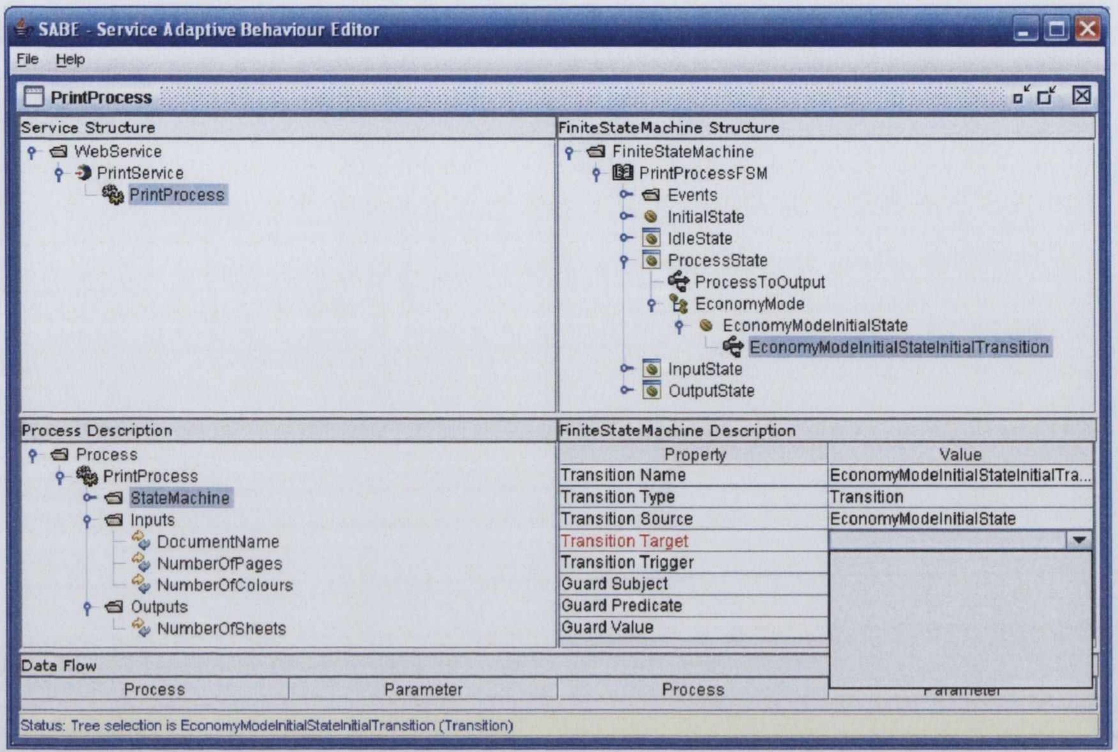


Property	Value
State Name	EconomyModelnitialState
State Type	PseudoState
State Kind	initial

Status: Tree selection is EconomyModelnitialState (PseudoState)

Figure 4-23, Adding a transition to the initial state

If you select this new transition, its details are loaded in the bottom right panel, showing that transition source is already configured, and required field transition target has an empty list since there is only this state (see Figure 4-24).


SABE - Service Adaptive Behaviour Editor

File Help

PrintProcess

Service Structure
- WebService
  - PrintService
    - PrintProcess

FiniteStateMachine Structure
- FiniteStateMachine
  - PrintProcessFSM
    - Events
    - InitialState
    - IdleState
    - ProcessState
      - ProcessToOutput
      - EconomyMode
        - EconomyModeInitialState
          - EconomyModeInitialStateInitialTransition
    - InputState
    - OutputState

Process Description
- Process
  - PrintProcess
    - StateMachine
    - Inputs
      - DocumentName
      - NumberOfPages
      - NumberOfColours
    - Outputs
      - NumberOfSheets

FiniteStateMachine Description

Property	Value
Transition Name	EconomyModeInitialStateInitialTra...
Transition Type	Transition
Transition Source	EconomyModeInitialState
Transition Target	
Transition Trigger	
Guard Subject	
Guard Predicate	
Guard Value	

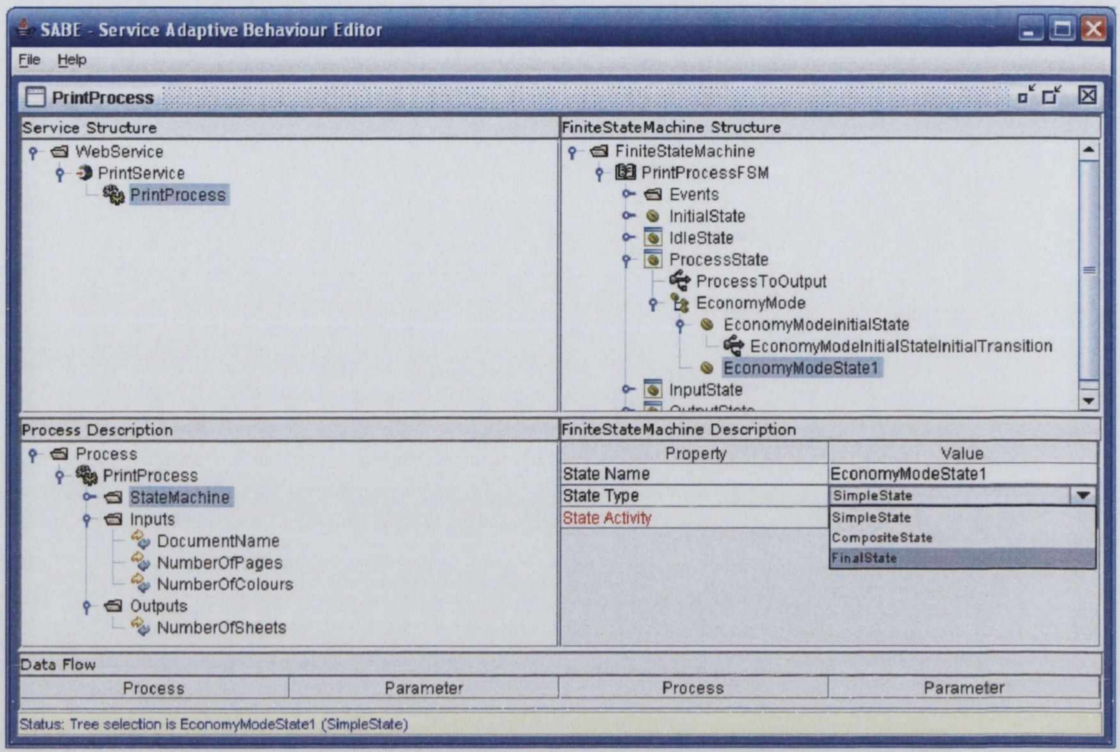
Data Flow

Process	Parameter	Process	Parameter

Status: Tree selection is EconomyModeInitialStateInitialTransition (Transition)

Figure 4-24, Missing aspects of the new transition are highlighted in red

vii. Another state must be added. Right click EconomyMode node and select the appropriate option. If this is the last state of the FSM, then this state should be a final state with an action (activity) which is a required property (see Figure 4-25).



SABE - Service Adaptive Behaviour Editor

File Help

PrintProcess

Service Structure

```
WebService
  PrintService
    PrintProcess
```

FiniteStateMachine Structure

```
FiniteStateMachine
  PrintProcessFSM
    Events
    InitialState
    IdleState
    ProcessState
      ProcessToOutput
      EconomyMode
        EconomyModeInitialState
          EconomyModeInitialStateInitialTransition
        EconomyModeState1
    InputState
```

Process Description

```
Process
  PrintProcess
    StateMachine
    Inputs
      DocumentName
      NumberOfPages
      NumberOfColours
    Outputs
      NumberOfSheets
```

FiniteStateMachine Description

Property	Value
State Name	EconomyModeState1
State Type	SimpleState
State Activity	SimpleState CompositeState FinalState

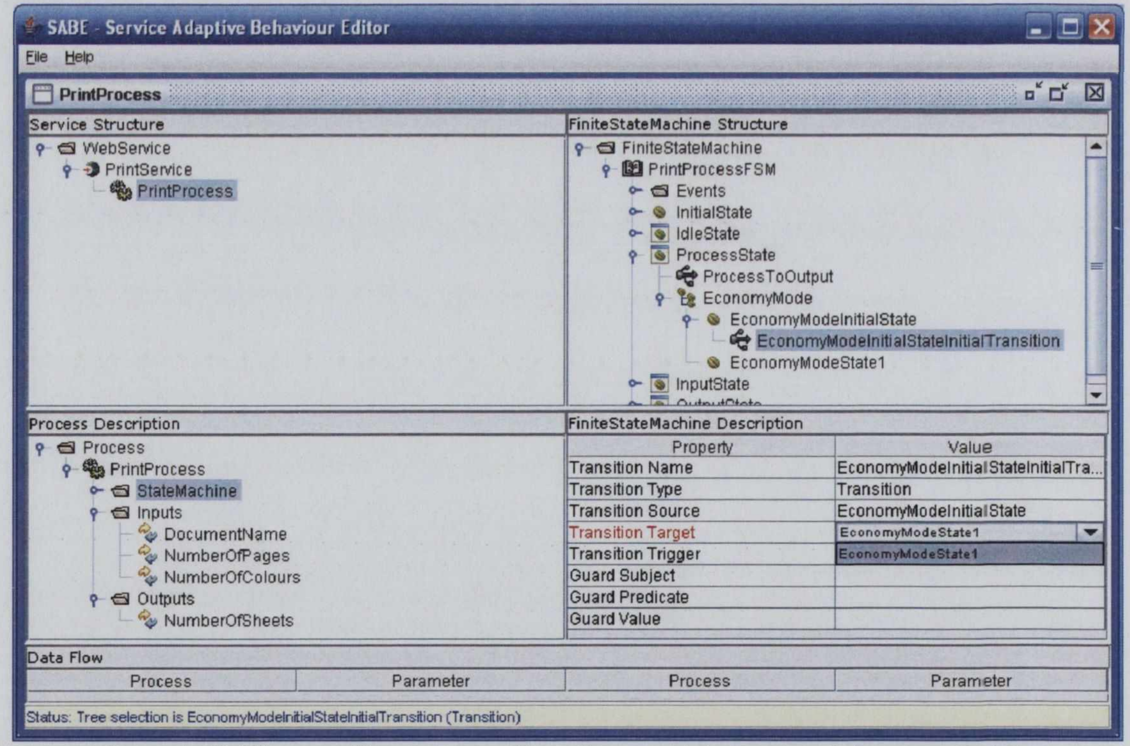
Data Flow

Process	Parameter	Process	Parameter

Status: Tree selection is EconomyModeState1 (SimpleState)

Figure 4-25, Adding a new state where state type options are provided

Now the transition from initial state can be configured by selecting the new state as its transition target (see Figure 4-26).



SABE - Service Adaptive Behaviour Editor

File Help

PrintProcess

Service Structure

```
WebService
  PrintService
    PrintProcess
```

FiniteStateMachine Structure

```
FiniteStateMachine
  PrintProcessFSM
    Events
    InitialState
    IdleState
    ProcessState
      ProcessToOutput
      EconomyMode
        EconomyModeInitialState
          EconomyModeInitialStateInitialTransition
        EconomyModeState1
    InputState
```

Process Description

```
Process
  PrintProcess
    StateMachine
    Inputs
      DocumentName
      NumberOfPages
      NumberOfColours
    Outputs
      NumberOfSheets
```

FiniteStateMachine Description

Property	Value
Transition Name	EconomyModeInitialStateInitialTra...
Transition Type	Transition
Transition Source	EconomyModeInitialState
Transition Target	EconomyModeState1
Transition Trigger	EconomyModeState1
Guard Subject	
Guard Predicate	
Guard Value	

Data Flow

Process	Parameter	Process	Parameter

Status: Tree selection is EconomyModeInitialStateInitialTransition (Transition)

Figure 4-26, When configuring transition, the new state is suggested as target state

viii. The description of this simple adaptive behaviour is now complete and can be saved. When saving, the FSM model is validated. In case the FSM model is incomplete or any of the required fields have not been configured, the SABE notifies the user.

It was demonstrated how the SABE tool can be used to describe the adaptive behaviours of atomic web services. More importantly, it demonstrates how the SABE tool can be used to accomplish the *Web Service Adaptivity Description Activity* of the MAWS methodology.

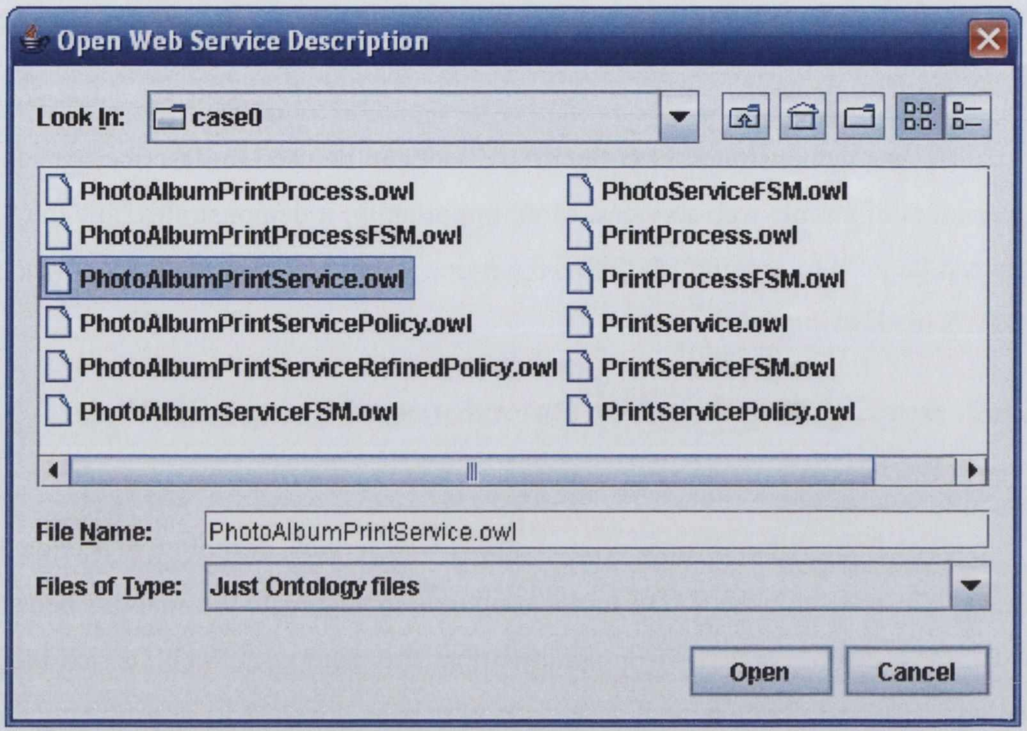
4.6.2 Aggregating Adaptive Behaviours of Composite Web Services

The PhotoAlbumPrintService example web service, described in section 3.3, is used to illustrate how the SABE tool is used here to aggregate the adaptive behaviours of composite web services. It is assumed that this composite web service has been semantically described, as well as the adaptive behaviours of its constituent adaptive web services. The steps for aggregating these adaptive behaviours are as follows:

i. Load the semantic description of the PhotoAlbumPrintService;
ii. Expand the PhotoAlbumPrintService node to see the service's composition;
iii. Select the PhotoAlbumPrintService node and add a FSM to represent the adaptive behaviours for this composite service;
iv. Review that the FSM created for this service contains the aggregated adaptive behaviours pertaining to its constituent adaptive web services;
v. Save the service's FSM;

Screenshots are used to visually aid in the sequence of actions performed on the SABE tool to describe such adaptive behaviour. Note that descriptions are above the screenshots.

i. Select the PhotoAlbumPrintService.owl file (see Figure 4-27).



caseO

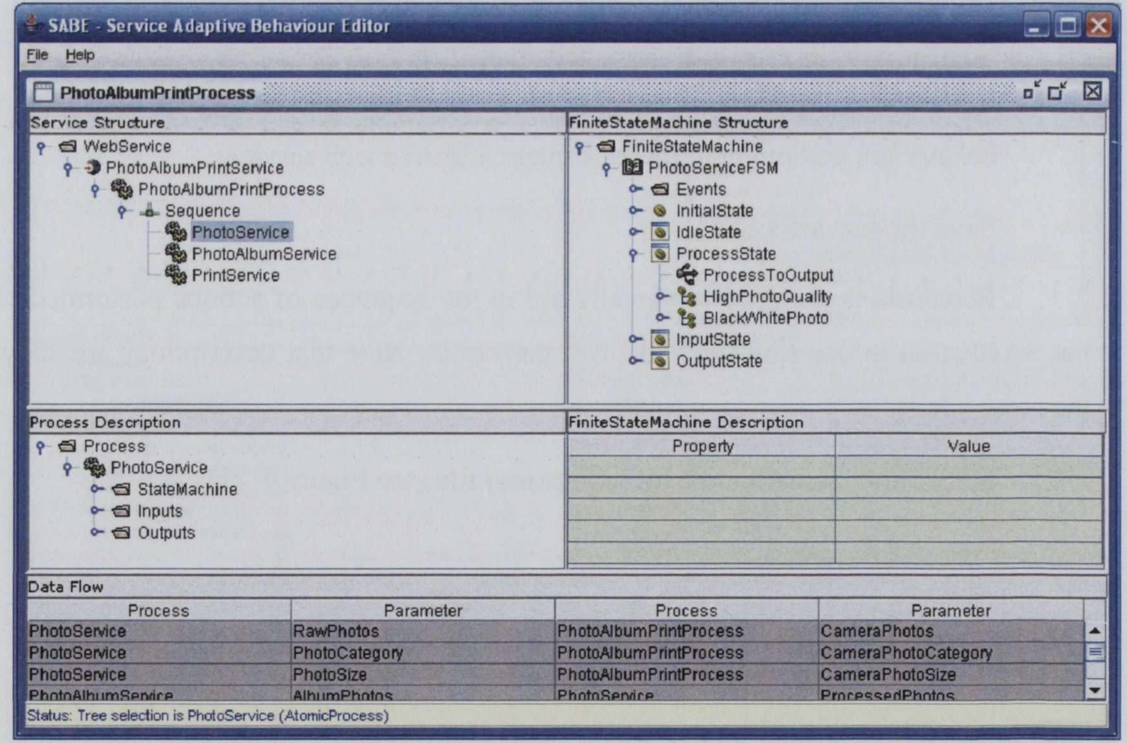
PhotoAlbumPrintProcess.owl	PhotoServiceFSM.owl
PhotoAlbumPrintProcessFSM.owl	PrintProcess.owl
PhotoAlbumPrintService.owl	PrintProcessFSM.owl
PhotoAlbumPrintServicePolicy.owl	PrintService.owl
PhotoAlbumPrintServiceRefinedPolicy.owl	PrintServiceFSM.owl
PhotoAlbumServiceFSM.owl	PrintServicePolicy.owl

File Name: PhotoAlbumPrintService.owl

Files of Type: Just Ontology files

Figure 4-27, Loading the description for composite service PhotoAlbumPrintService

ii. Expand the PhotoAlbumPrintService node on the top left panel to see its composition structure. By selecting each of its constituent atomic web services, description of their adaptive behaviours can be seeing on the top right panel (see Figure 4-28).



Service Structure

```
WebService
  PhotoAlbumPrintService
    PhotoAlbumPrintProcess
      Sequence
        PhotoService
        PhotoAlbumService
        PrintService
```

FiniteStateMachine Structure

```
FiniteStateMachine
  PhotoServiceFSM
    Events
    InitialState
    IdleState
    ProcessState
      ProcessToOutput
      HighPhotoQuality
      BlackWhitePhoto
    InputState
    OutputState
```

Process Description

```
Process
  PhotoService
    StateMachine
    Inputs
    Outputs
```

FiniteStateMachine Description

Property	Value

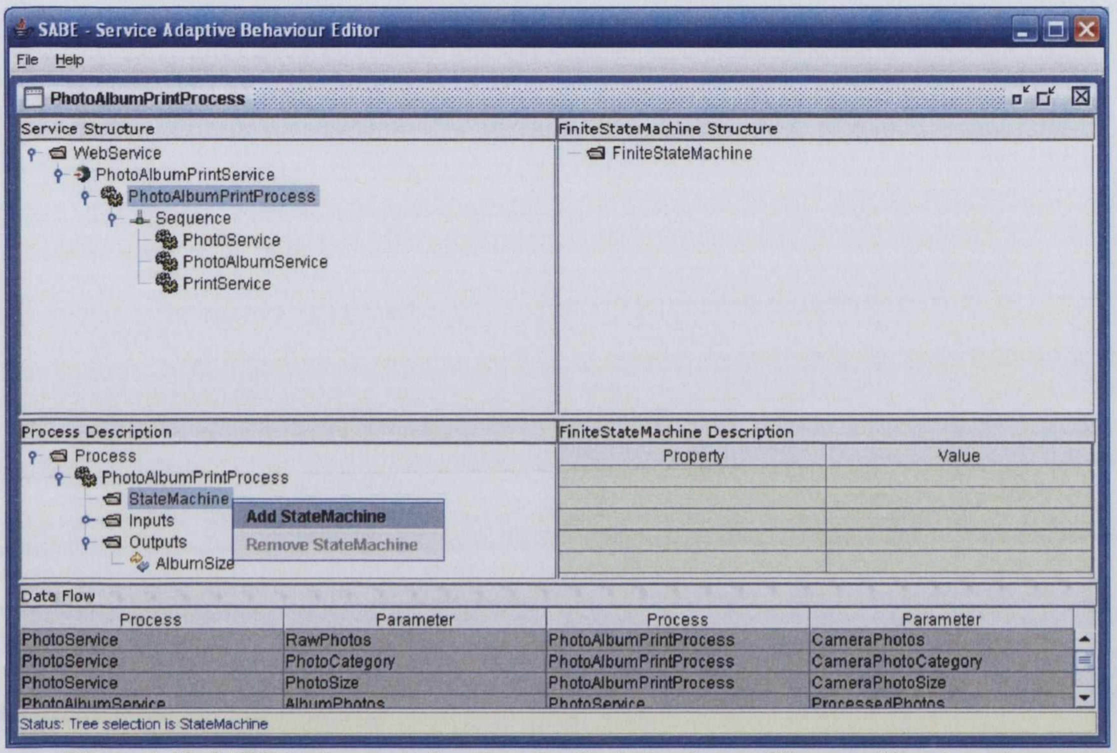
Data Flow

Process	Parameter	Process	Parameter
PhotoService	RawPhotos	PhotoAlbumPrintProcess	CameraPhotos
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	CameraPhotoCategory
PhotoService	PhotoSize	PhotoAlbumPrintProcess	CameraPhotoSize
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos

Status: Tree selection is PhotoService (AtomicProcess)

Figure 4-28, SABE tool displaying the adaptive behaviours of one of the constituent services

iii. Select PhotoAlbumPrintProcess, and the process details of this service are loaded in the bottom left panel. Expand it, and right click on the StateMachine node to see a dropdown menu with an option to add a StateMachine (FSM). Select this option to create a FSM and automatically aggregate the adaptive behaviours of its constituent web services (see Figure 4-29).

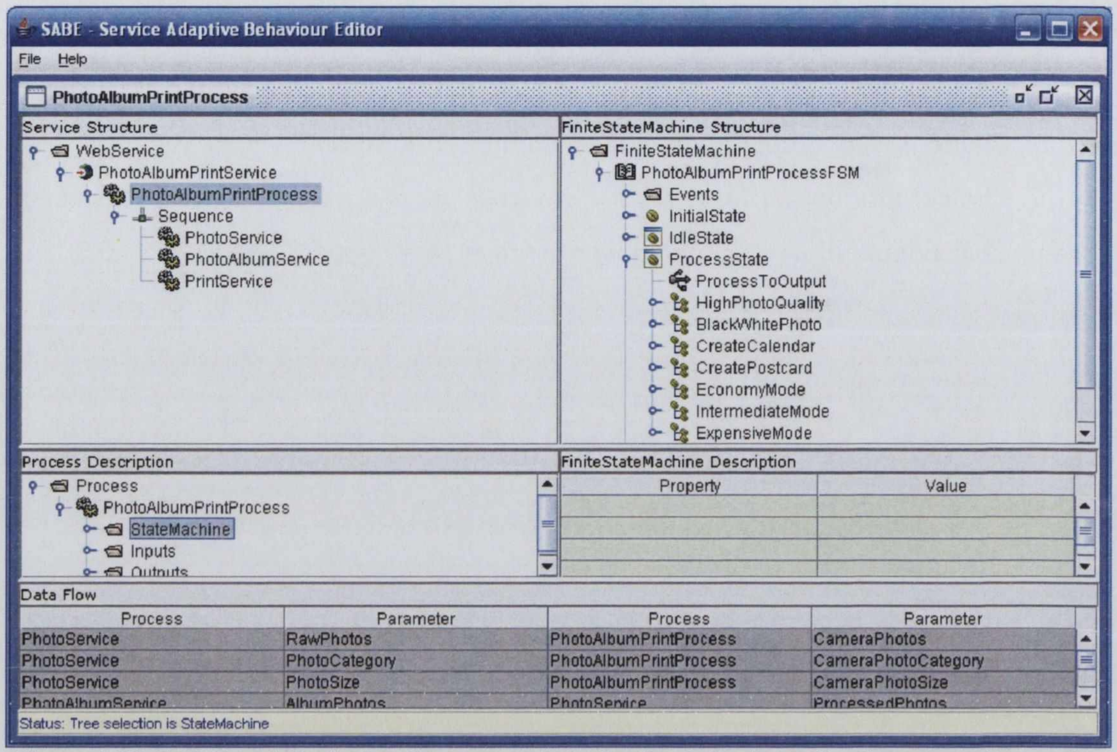


Process	Parameter	Process	Parameter
PhotoService	RawPhotos	PhotoAlbumPrintProcess	CameraPhotos
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	CameraPhotoCategory
PhotoService	PhotoSize	PhotoAlbumPrintProcess	CameraPhotoSize
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos

Status: Tree selection is StateMachine

Figure 4-29, SABE tool aggregating the adaptive behaviours of the constituent services

iv. Expand the FSM description on the top right panel to see that all the FSM describing the adaptive behaviours of its constituent web services have been aggregated and are included here. Note that the tool prevents the description of these FSM from being edited (see Figure 4-30).



Process	Parameter	Process	Parameter
PhotoService	RawPhotos	PhotoAlbumPrintProcess	CameraPhotos
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	CameraPhotoCategory
PhotoService	PhotoSize	PhotoAlbumPrintProcess	CameraPhotoSize
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos

Status: Tree selection is StateMachine

Figure 4-30, SABE tool displaying the aggregated adaptive behaviours for the composite service

v. The aggregation of the adaptive behaviours of this composite web service is now complete and can be saved. This is achieved by clicking File, then Save. When saving, the FSM model is validated. In case the FSM model is incomplete or any of the required fields have not been configured, the SABE notifies the user.

It was demonstrated how the SABE tool can be used to accomplish the *Web Service Adaptivity Aggregation Activity* of the MAWS methodology by aggregating the adaptive behaviours of composite web services in an automated manner.

4.7 Walkthrough in Using SMPE Tool to Create Policies to Manage Adaptive Composite Services

The SMPE tool was designed and implemented to allow administrators to specify high-level management policies which are automatically refined into auto-generated low-level policies that manage adaptive composite services. This chapter presented a lengthy an in-depth description of the design and implementation of this tool. However, the SMPE tool was designed with a GUI that shields users from the semantic description, and policy model in their lexical form. This tool also provides an automated policy refinement feature that facilitates managers in generating policies to manage adaptive composite services.

This section provides a work example of how web service managers can use the SMPE tool to specify and automatically refine management policies to manage adaptive composite services. This walkthrough should illustrate how this tool facilitates the users in creating policies to manage adaptive composite services. It provides illustrated example of how the SMPE tool supports the *Management Policy Specification Activity*, *Management Policy Refinement Activity* and *Management Policy Generation Activity* of the MAWS methodology.

4.7.1 Specifying Management Policies for Managing Adaptive Composite Web Services

The PhotoAlbumPrintService example web service, described earlier in section 3.3, is used here to illustrate how the SMPE tool is used. For the example, a management policy (shown in Table 3) was specified to: "Produce a colour photo calendar of the members of the U2 band".

PhotoAlbumPrint Service (CompositeProcess)			
Name	**Event**	**Condition**	**Action**
Policy1	ProcessEvent	PhotoCategory= = Portrait	Colour-Printing && Create-Calendar

Table 3, Another management policy for PhotoAlbumPrintService

Furthermore, it also depicts how this tool is used to refine this management policy. It is assumed that this adaptive web service has been implemented, and syntactically and semantically described. The steps for specifying this management policy and then refining it are as follows:

i. Start the SMPE tool;

ii. Load the PhotoAlbumPrintService from file PhotoAlbumPrintService.owl;

iii. Select the PhotoAlbumPrintService node and create a new policy for it;

iv. Expand the newly created management policy for this service;

v. Add an event for this policy;

vi. Add a condition for this management policy;

vii. Add a complex action (made of two actions) for this management policy;

viii. Configure first simple action;

ix. Configure the second action;

x. Save the policies created for this service;

Screenshots are used to visually aid in the sequence of actions performed on the SMPE tool to specify these management policies and refine them. Note that descriptions are above the screenshots.

i. Start the tool and select open (see Figure 4-31).

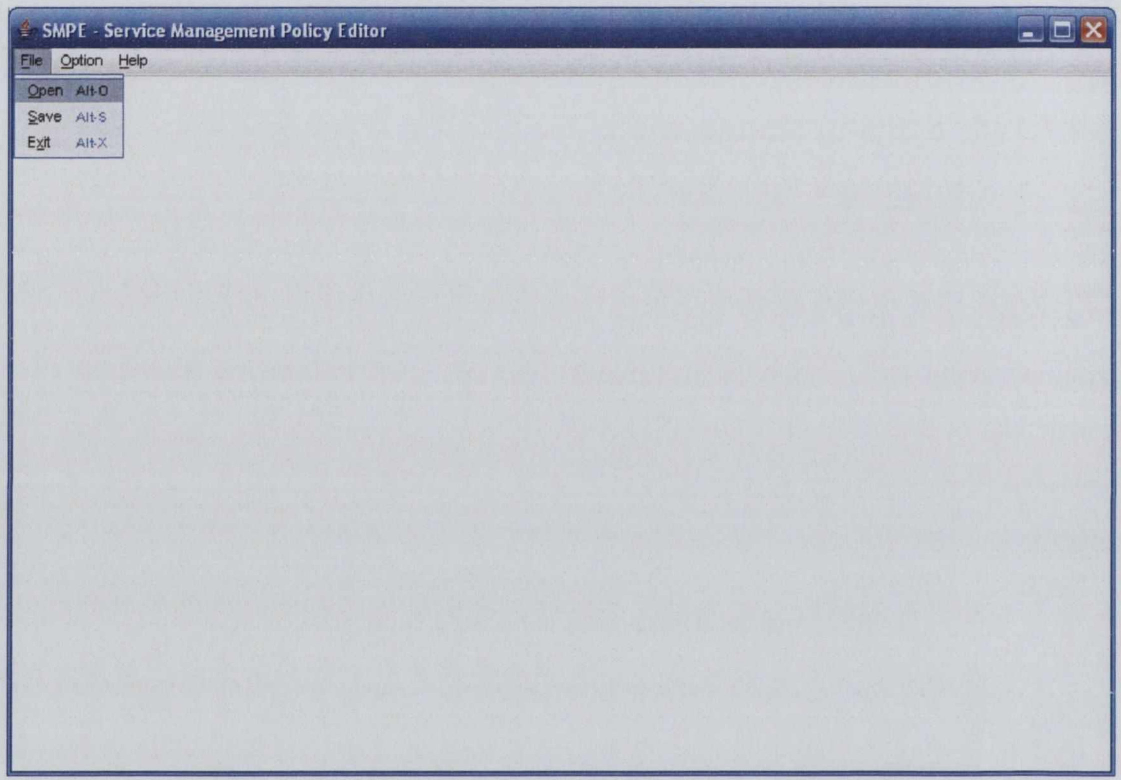
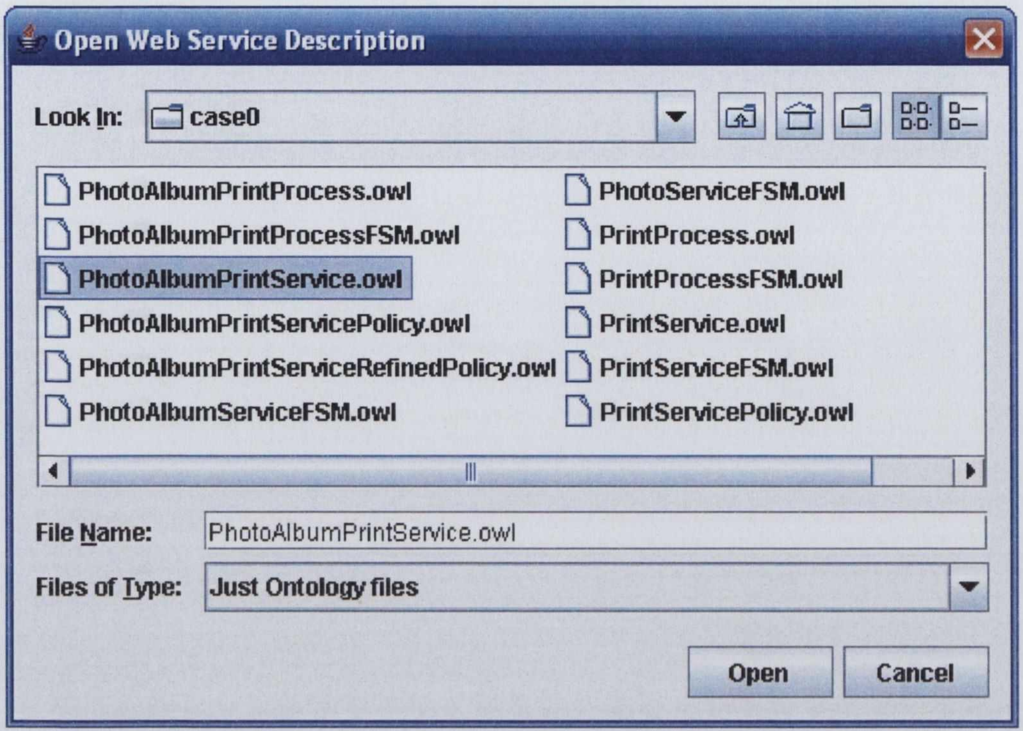


Figure 4-31, SMPE tool prior to loading web service description

ii. Open the PhotoAlbumPrintService.owl file (see Figure 4-32).



Open Web Service Description

Look In: case0

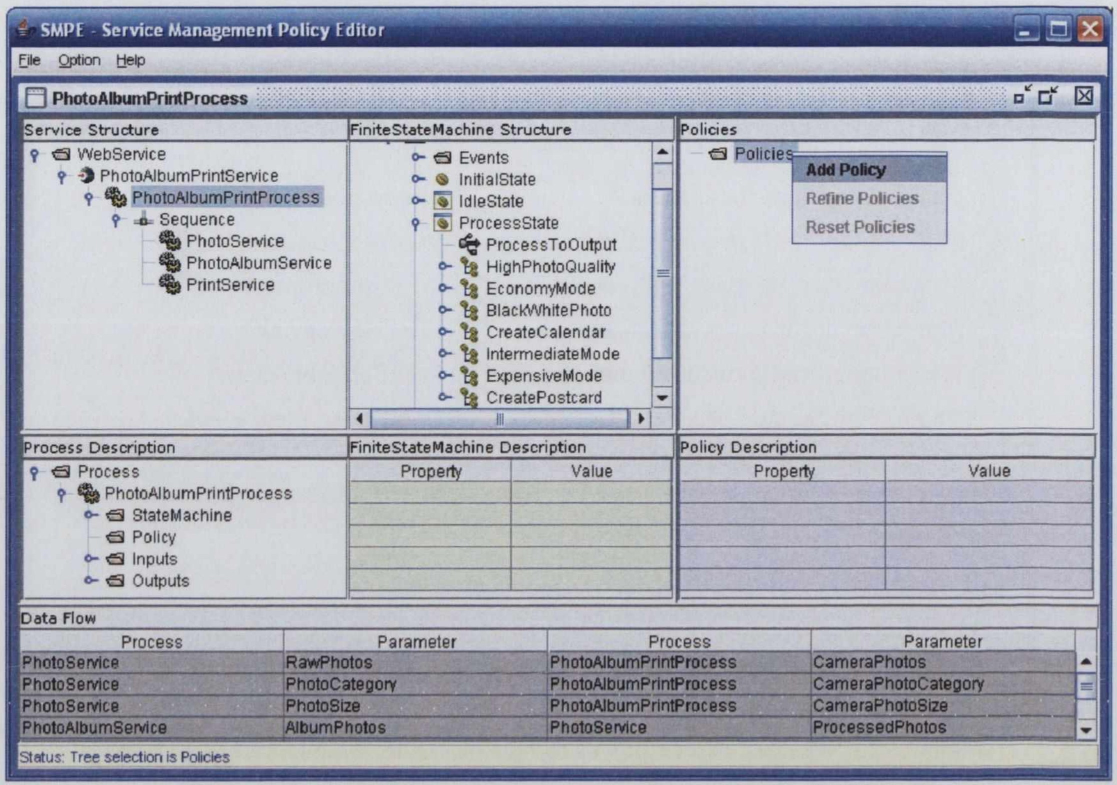
PhotoAlbumPrintProcess.owl	PhotoServiceFSM.owl
PhotoAlbumPrintProcessFSM.owl	PrintProcess.owl
PhotoAlbumPrintService.owl	PrintProcessFSM.owl
PhotoAlbumPrintServicePolicy.owl	PrintService.owl
PhotoAlbumPrintServiceRefinedPolicy.owl	PrintServiceFSM.owl
PhotoAlbumServiceFSM.owl	PrintServicePolicy.owl

File Name: PhotoAlbumPrintService.owl

Files of Type: Just Ontology files

Figure 4-32, Selecting description of composite service PhotoAlbumPrintService

iii. The left side of this tool is similar to the SABE tool. Policies can only be specified to the topmost web service, in this case the PhotoAlbumPrint Service. Hence, select the PhotoAlbumPrintProcess from the top left panel and the other panels will load with its relevant details. Right click on Policies node on the top right panel to add a management policy (see Figure 4-33).

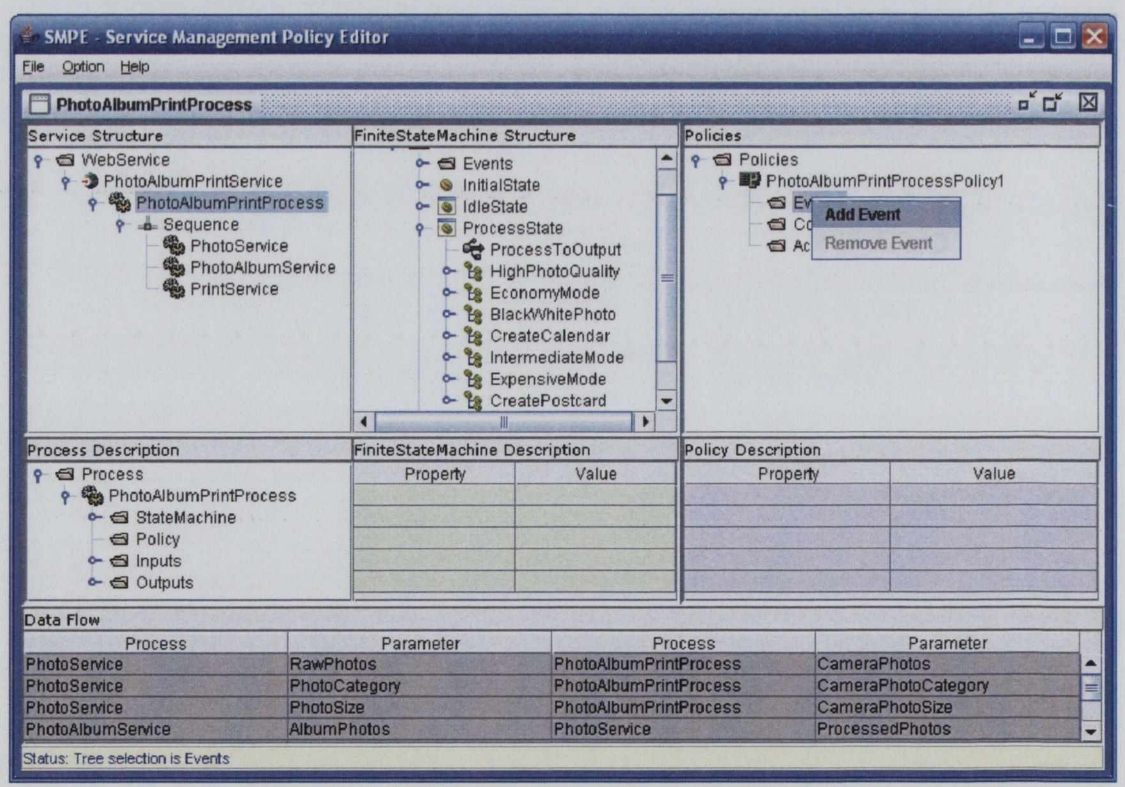


Process	Parameter	Process	Parameter
PhotoService	RawPhotos	PhotoAlbumPrintProcess	CameraPhotos
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	CameraPhotoCategory
PhotoService	PhotoSize	PhotoAlbumPrintProcess	CameraPhotoSize
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos

Status: Tree selection is Policies

Figure 4-33, Specifying a management policy using the SMPE tool

iv. Expand the new policy and see that it has three aspects: event, condition, and action. Right click on the event node to add an event (see Figure 4-34).

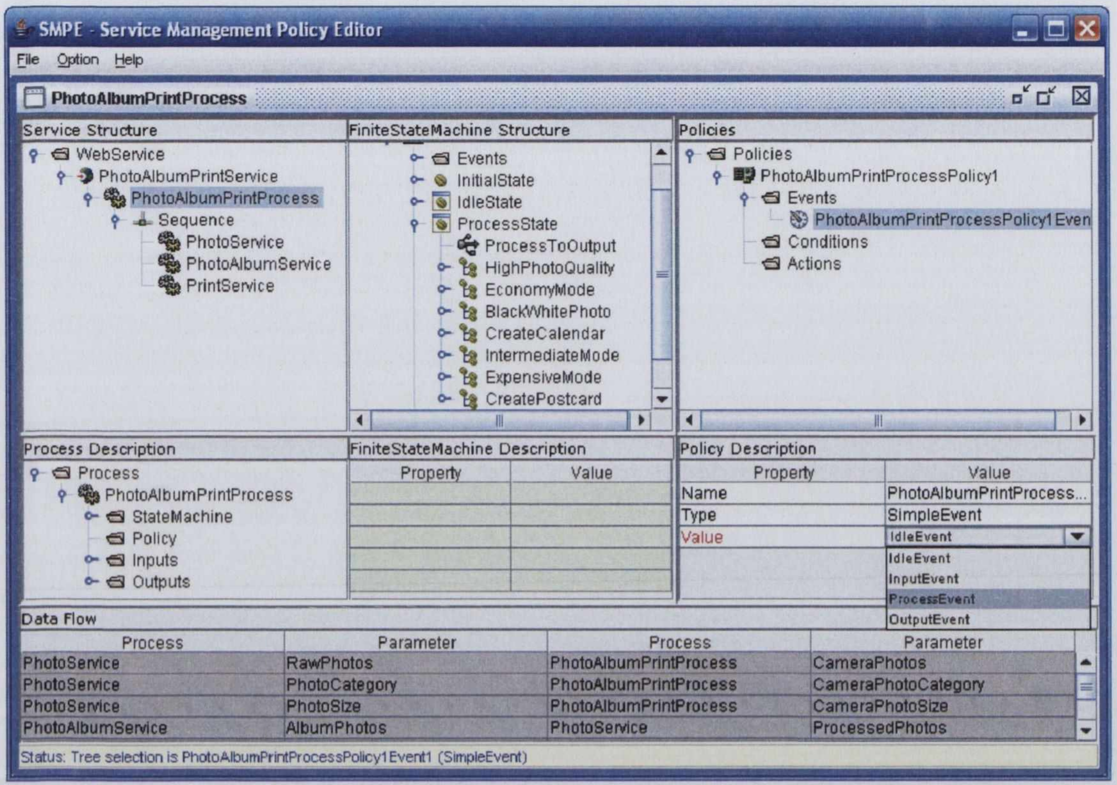


Process	Parameter	Process	Parameter
PhotoService	RawPhotos	PhotoAlbumPrintProcess	CameraPhotos
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	CameraPhotoCategory
PhotoService	PhotoSize	PhotoAlbumPrintProcess	CameraPhotoSize
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos

Status: Tree selection is Events

Figure 4-34, Adding an event to the management policy

v. Select the new node and configure its properties in the bottom right panel. Required fields are highlighted in red. Select the appropriate event from a list of events (see Figure 4-35).


SMPE - Service Management Policy Editor

File Option Help

PhotoAlbumPrintProcess

Service Structure

```
WebService
  PhotoAlbumPrintService
    PhotoAlbumPrintProcess
      Sequence
        PhotoService
        PhotoAlbumService
        PrintService
```

FiniteStateMachine Structure

```
Events
InitialState
IdleState
ProcessState
  ProcessToOutput
  HighPhotoQuality
  EconomyMode
  BlackWhitePhoto
  CreateCalendar
  IntermediateMode
  ExpensiveMode
  CreatePostcard
```

Policies

```
Policies
  PhotoAlbumPrintProcessPolicy1
    Events
      PhotoAlbumPrintProcessPolicy1 Even
    Conditions
    Actions
```

Process Description

```
Process
  PhotoAlbumPrintProcess
    StateMachine
    Policy
    Inputs
    Outputs
```

FiniteStateMachine Description

Property	Value

Policy Description

Property	Value
Name	PhotoAlbumPrintProcess...
Type	SimpleEvent
Value	IdleEvent

IdleEvent
InputEvent
ProcessEvent
OutputEvent

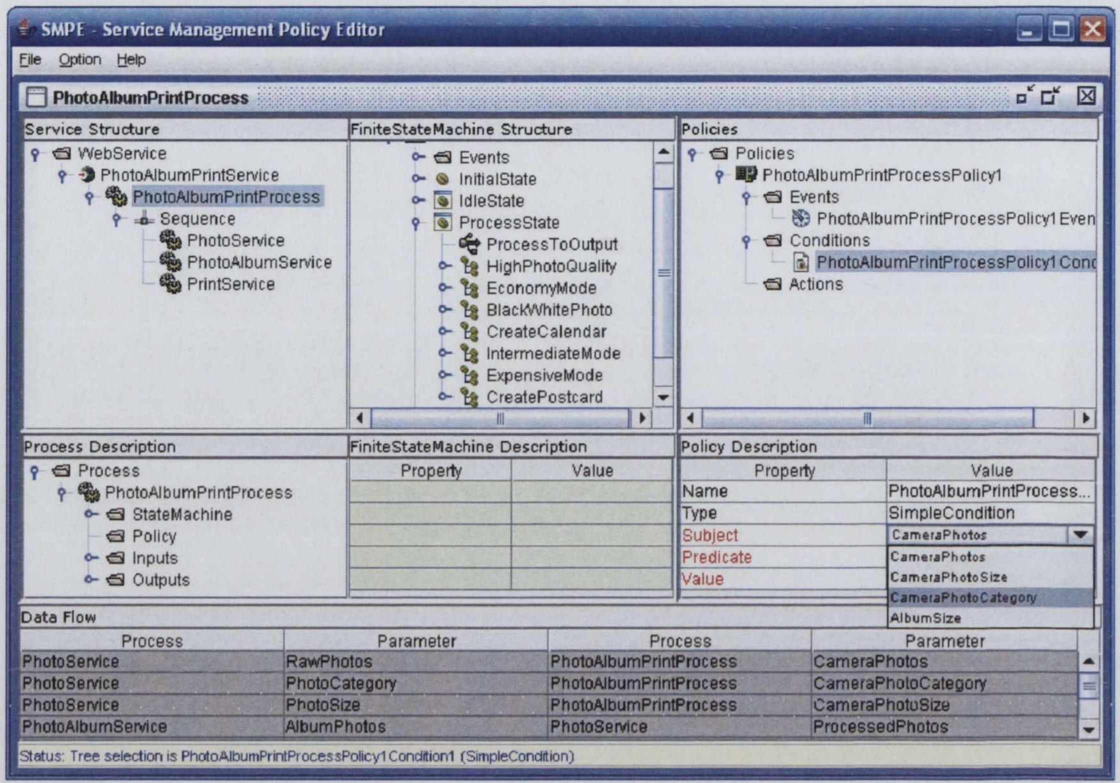
Data Flow

Process	Parameter	Process	Parameter
PhotoService	RawPhotos	PhotoAlbumPrintProcess	CameraPhotos
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	CameraPhotoCategory
PhotoService	PhotoSize	PhotoAlbumPrintProcess	CameraPhotoSize
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos

Status: Tree selection is PhotoAlbumPrintProcessPolicy1Event1 (SimpleEvent)

Figure 4-35, SMPE tool highlights in red the missing details and provides options

vi. Add a new condition and configure its properties in the bottom right panel. Required fields are highlighted in red. A list of the service's parameters is available for the subject property and a list of predicates (equal to, not equal to, less than, greater than) for the predicate property (see Figure 4-36).



Service Structure

```
WebService
  PhotoAlbumPrintService
    PhotoAlbumPrintProcess
      Sequence
        PhotoService
        PhotoAlbumService
        PrintService
```

FiniteStateMachine Structure

```
Events
InitialState
IdleState
ProcessState
  ProcessToOutput
  HighPhotoQuality
  EconomyMode
  BlackWhitePhoto
  CreateCalendar
  IntermediateMode
  ExpensiveMode
  CreatePostcard
```

Policies

```
Policies
  PhotoAlbumPrintProcessPolicy1
    Events
      PhotoAlbumPrintProcessPolicy1Even
    Conditions
      PhotoAlbumPrintProcessPolicy1Conc
    Actions
```

Process Description

```
Process
  PhotoAlbumPrintProcess
    StateMachine
    Policy
    Inputs
    Outputs
```

FiniteStateMachine Description

Property	Value

Policy Description

Property	Value
Name	PhotoAlbumPrintProcess...
Type	SimpleCondition
Subject	CameraPhotos
Predicate	
Value	

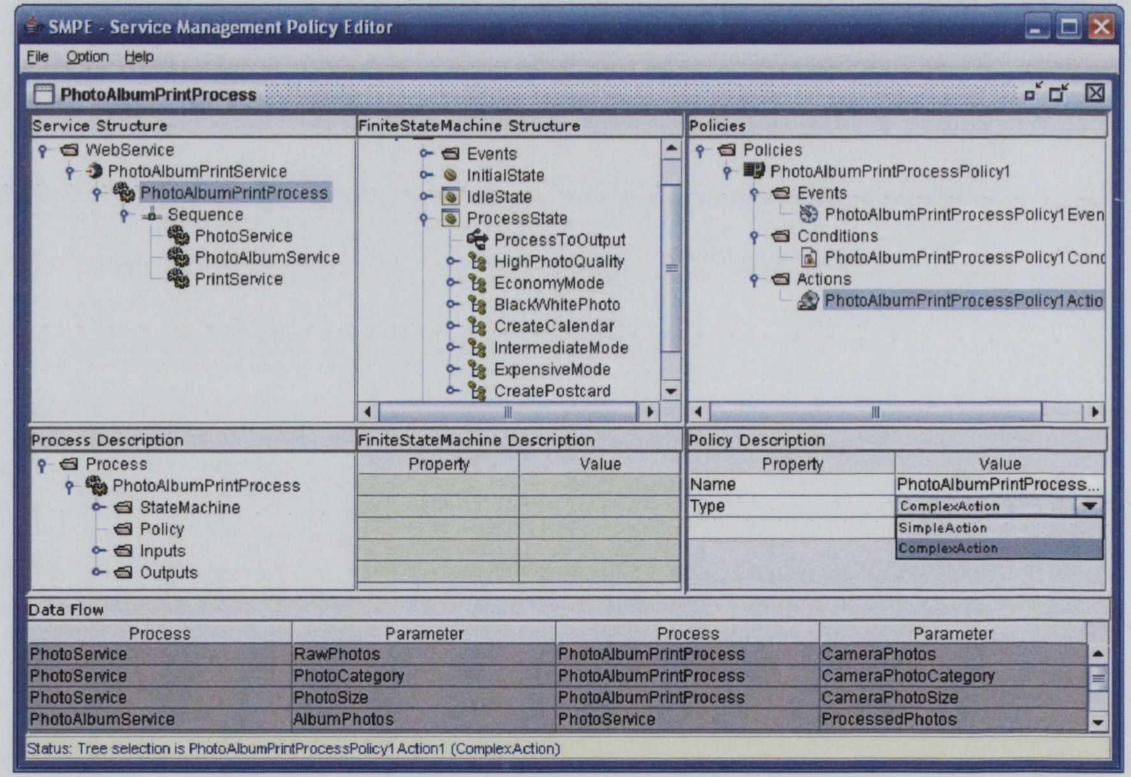
Data Flow

Process	Parameter	Process	Parameter
PhotoService	RawPhotos	PhotoAlbumPrintProcess	CameraPhotos
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	CameraPhotoCategory
PhotoService	PhotoSize	PhotoAlbumPrintProcess	CameraPhotoSize
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos

Status: Tree selection is PhotoAlbumPrintProcessPolicy1Condition1 (SimpleCondition)

Figure 4-36, Selecting subject for new condition from selection of service's parameters

vii. Add an action and configure its type attribute to be ComplexAction, since this policy will contain more than one action (see Figure 4-37).



Service Structure

```
WebService
  PhotoAlbumPrintService
    PhotoAlbumPrintProcess
      Sequence
        PhotoService
        PhotoAlbumService
        PrintService
```

FiniteStateMachine Structure

```
Events
InitialState
IdleState
ProcessState
  ProcessToOutput
  HighPhotoQuality
  EconomyMode
  BlackWhitePhoto
  CreateCalendar
  IntermediateMode
  ExpensiveMode
  CreatePostcard
```

Policies

```
Policies
  PhotoAlbumPrintProcessPolicy1
    Events
      PhotoAlbumPrintProcessPolicy1Even
    Conditions
      PhotoAlbumPrintProcessPolicy1Conc
    Actions
      PhotoAlbumPrintProcessPolicy1Actio
```

Process Description

```
Process
  PhotoAlbumPrintProcess
    StateMachine
    Policy
    Inputs
    Outputs
```

FiniteStateMachine Description

Property	Value

Policy Description

Property	Value
Name	PhotoAlbumPrintProcess...
Type	ComplexAction

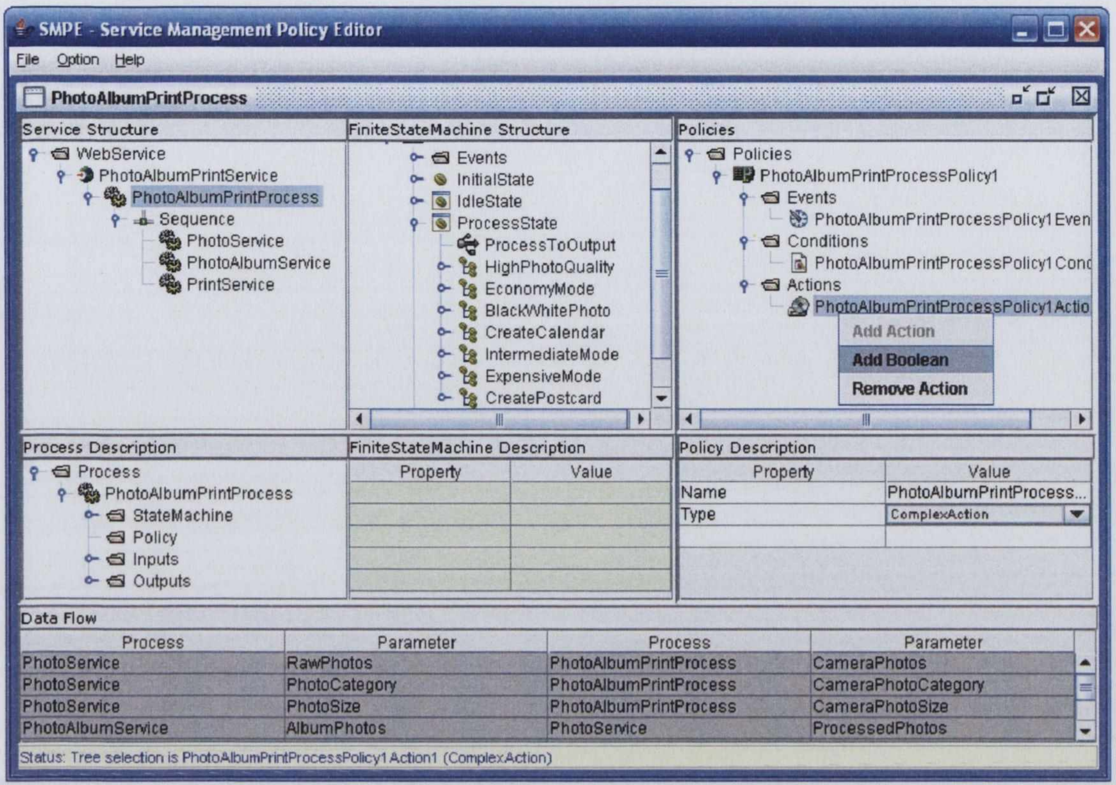
Data Flow

Process	Parameter	Process	Parameter
PhotoService	RawPhotos	PhotoAlbumPrintProcess	CameraPhotos
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	CameraPhotoCategory
PhotoService	PhotoSize	PhotoAlbumPrintProcess	CameraPhotoSize
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos

Status: Tree selection is PhotoAlbumPrintProcessPolicy1Action1 (ComplexAction)

Figure 4-37, Adding a complex action to management policy

Add a Boolean node for this complex action (see Figure 4-38).



Policy Description

Property	Value
Name	PhotoAlbumPrintProcess...
Type	ComplexAction

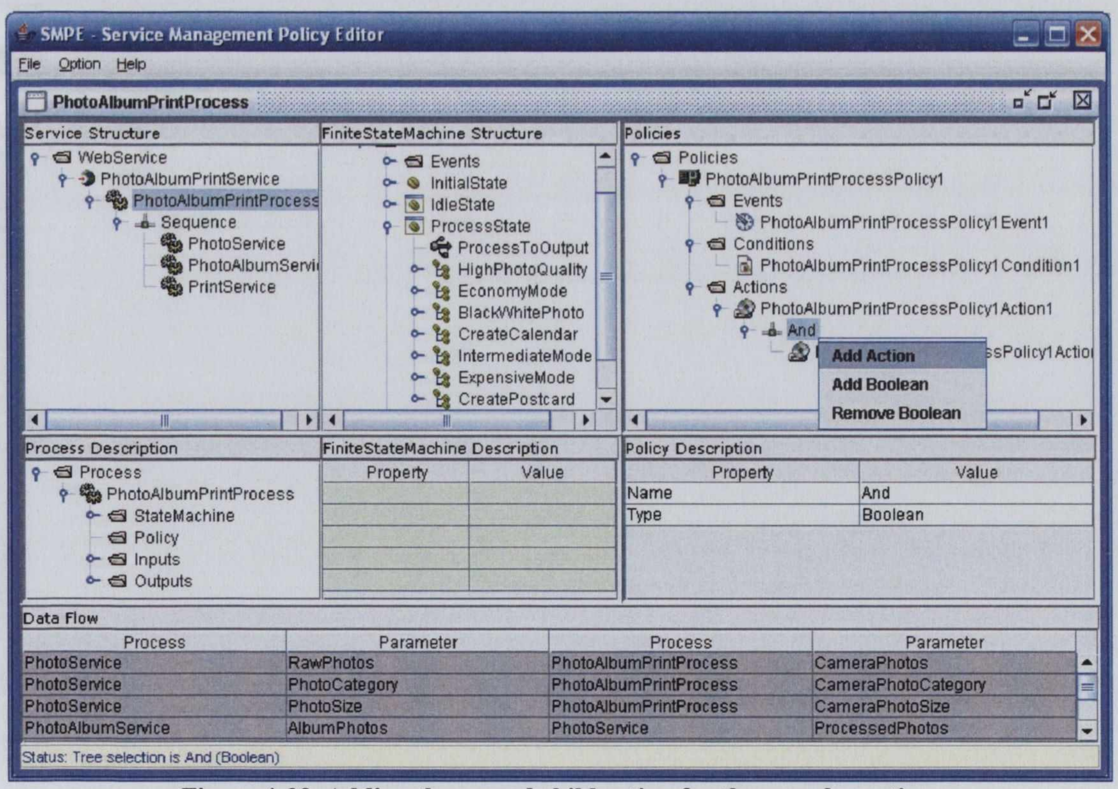
Data Flow

Process	Parameter	Process	Parameter
PhotoService	RawPhotos	PhotoAlbumPrintProcess	CameraPhotos
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	CameraPhotoCategory
PhotoService	PhotoSize	PhotoAlbumPrintProcess	CameraPhotoSize
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos

Status: Tree selection is PhotoAlbumPrintProcessPolicy1Action1 (ComplexAction)

Figure 4-38, Adding a Boolean node for the complex action

It is configured to be AND by default. Then add two action nodes (see Figure 4-39).



Policy Description

Property	Value
Name	And
Type	Boolean

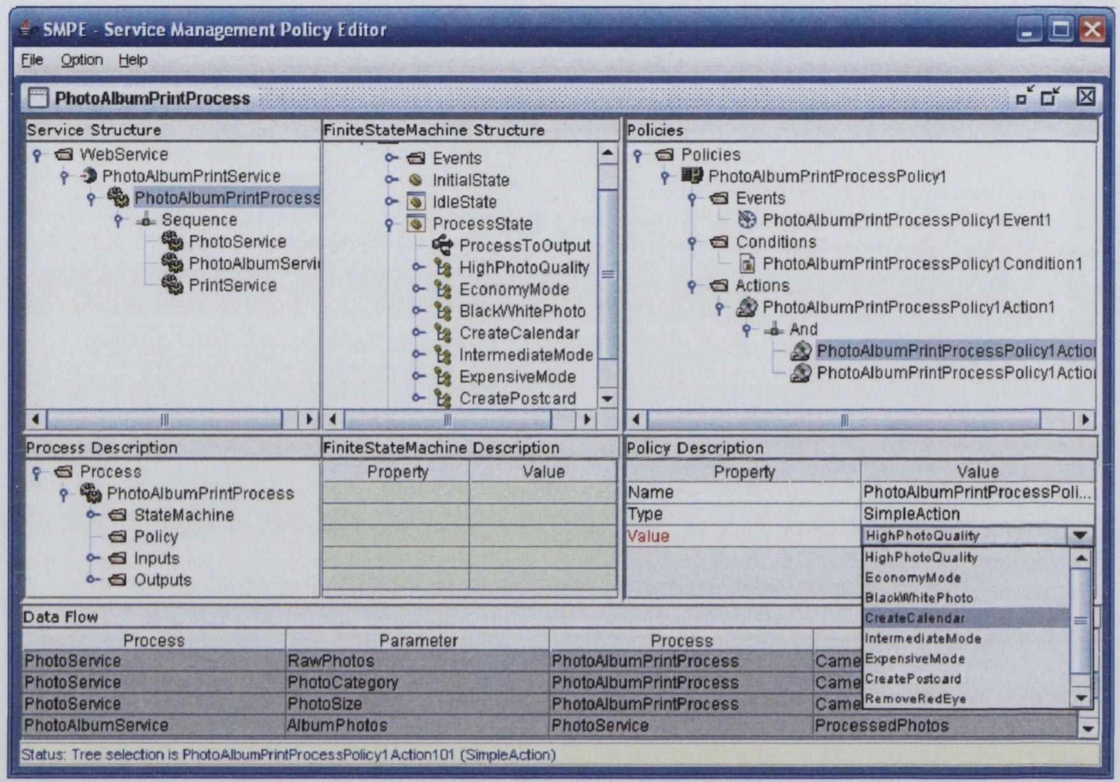
Data Flow

Process	Parameter	Process	Parameter
PhotoService	RawPhotos	PhotoAlbumPrintProcess	CameraPhotos
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	CameraPhotoCategory
PhotoService	PhotoSize	PhotoAlbumPrintProcess	CameraPhotoSize
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos

Status: Tree selection is And (Boolean)

Figure 4-39, Adding the second child action for the complex action

viii. Select the first new action node and its properties will load in the bottom right panel. The value property for this action is highlighted in red since it is a required property. Choose an appropriate action value from the list of adaptive behaviours; in this case it is CreateCalendar (see Figure 4-40).



SMPE - Service Management Policy Editor

File Option Help

PhotoAlbumPrintProcess

Service Structure
- WebService
  - PhotoAlbumPrintService
    - PhotoAlbumPrintProcess
      - Sequence
        - PhotoService
        - PhotoAlbumServi
        - PrintService

FiniteStateMachine Structure
- Events
- InitialState
- IdleState
- ProcessState
  - ProcessToOutput
  - HighPhotoQuality
  - EconomyMode
  - BlackWhitePhoto
  - CreateCalendar
  - IntermediateMode
  - ExpensiveMode
  - CreatePostcard

Policies
- Policies
  - PhotoAlbumPrintProcessPolicy1
    - Events
      - PhotoAlbumPrintProcessPolicy1Event1
    - Conditions
      - PhotoAlbumPrintProcessPolicy1Condition1
    - Actions
      - PhotoAlbumPrintProcessPolicy1Action1
        - And
          - PhotoAlbumPrintProcessPolicy1Actio
          - PhotoAlbumPrintProcessPolicy1Actio

Process Description
- Process
  - PhotoAlbumPrintProcess
    - StateMachine
    - Policy
    - Inputs
    - Outputs

FiniteStateMachine Description

Property	Value

Policy Description

Property	Value
Name	PhotoAlbumPrintProcessPoli...
Type	SimpleAction
Value	HighPhotoQuality

HighPhotoQuality
EconomyMode
BlackWhitePhoto
CreateCalendar
IntermediateMode
ExpensiveMode
CreatePostcard
RemoveRedEye

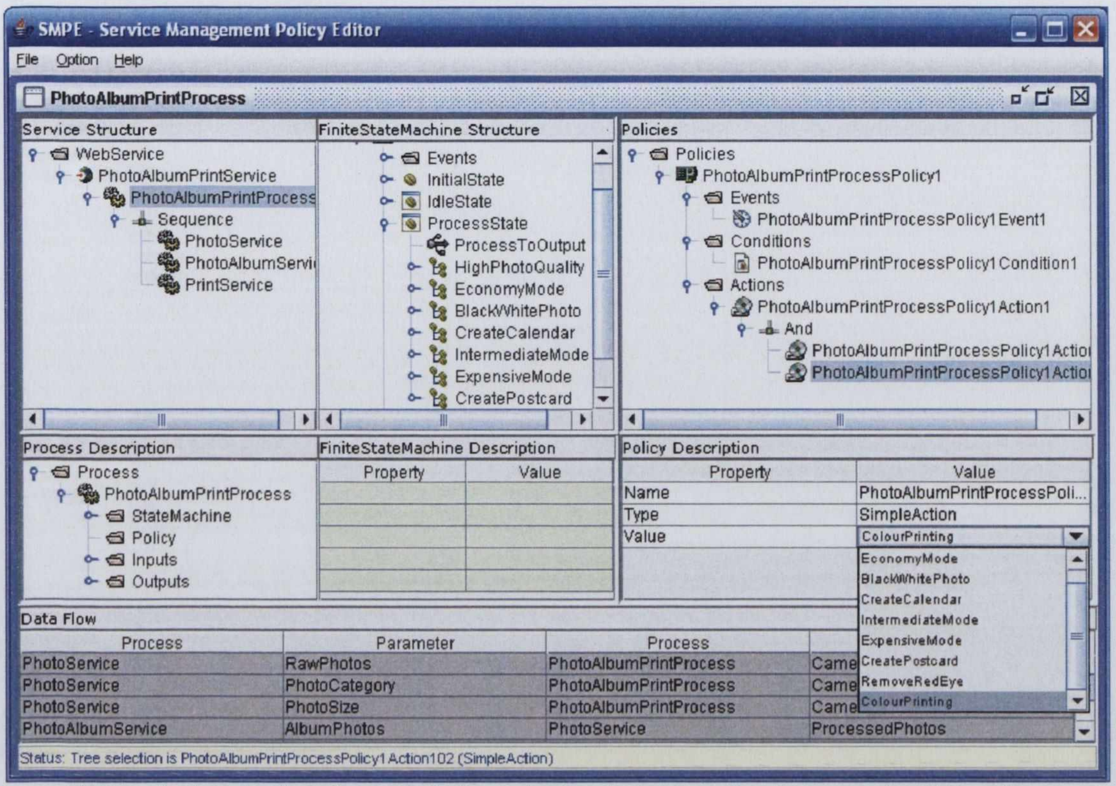
Data Flow

Process	Parameter	Process	
PhotoService	RawPhotos	PhotoAlbumPrintProcess	Came
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	Came
PhotoService	PhotoSize	PhotoAlbumPrintProcess	Came
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos

Status: Tree selection is PhotoAlbumPrintProcessPolicy1Action101 (SimpleAction)

Figure 4-40, Configuring the first policy action from dropdown menu to be CreateCalendar

ix. Configure the second node to have an action value of ColourPrinting (see Figure 4-41).


SMPE - Service Management Policy Editor

File Option Help

PhotoAlbumPrintProcess

Service Structure

```
WebService
  PhotoAlbumPrintService
    PhotoAlbumPrintProcess
      Sequence
        PhotoService
        PhotoAlbumServi
        PrintService
```

FiniteStateMachine Structure

```
Events
InitialState
IdleState
ProcessState
  ProcessToOutput
  HighPhotoQuality
  EconomyMode
  BlackWhitePhoto
  CreateCalendar
  IntermediateMode
  ExpensiveMode
  CreatePostcard
```

Policies

```
Policies
  PhotoAlbumPrintProcessPolicy1
    Events
      PhotoAlbumPrintProcessPolicy1Event1
    Conditions
      PhotoAlbumPrintProcessPolicy1Condition1
    Actions
      PhotoAlbumPrintProcessPolicy1Action1
        And
          PhotoAlbumPrintProcessPolicy1Actio
          PhotoAlbumPrintProcessPolicy1Actio
```

Process Description

```
Process
  PhotoAlbumPrintProcess
    StateMachine
    Policy
    Inputs
    Outputs
```

FiniteStateMachine Description

Property	Value

Policy Description

Property	Value
Name	PhotoAlbumPrintProcessPoli...
Type	SimpleAction
Value	ColourPrinting

```
EconomyMode
BlackWhitePhoto
CreateCalendar
IntermediateMode
ExpensiveMode
CreatePostcard
RemoveRedEye
ColourPrinting
```

Data Flow

Process	Parameter	Process	
PhotoService	RawPhotos	PhotoAlbumPrintProcess	Came
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	Came
PhotoService	PhotoSize	PhotoAlbumPrintProcess	Came
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos

Status: Tree selection is PhotoAlbumPrintProcessPolicy1Action102 (SimpleAction)

Figure 4-41, Selecting ColourPrinting adaptive behaviour for the second policy action

x. The specification of this high-level management policy for the adaptive composite service PhotoAlbumPrintService is now complete and can be saved. This is achieved by clicking File, then Save. When saving, management policies specified are validated. The SMPE notifies the user in case any of the specified policies are incomplete or any of the required fields have not been configured.

It was shown how the SMPE tool can be used to accomplish the Management Policy Specification Activity of the MAWS methodology by allowing user to specify management policies for adaptive (composite) services.

4.7.2 Automatically Refining Management Policies for Adaptive Composite Web Services

The PhotoAlbumPrint Service example web service is used here to illustrate how the SMPE tool automatically refines the specified management policies. It is assumed that management policies have been specified for this service. The refinement process produces low-level enforceable policies for the relevant constituent adaptive web services. The steps for refining this management policy are as follows:

i. Refine the specified management policy for this service;

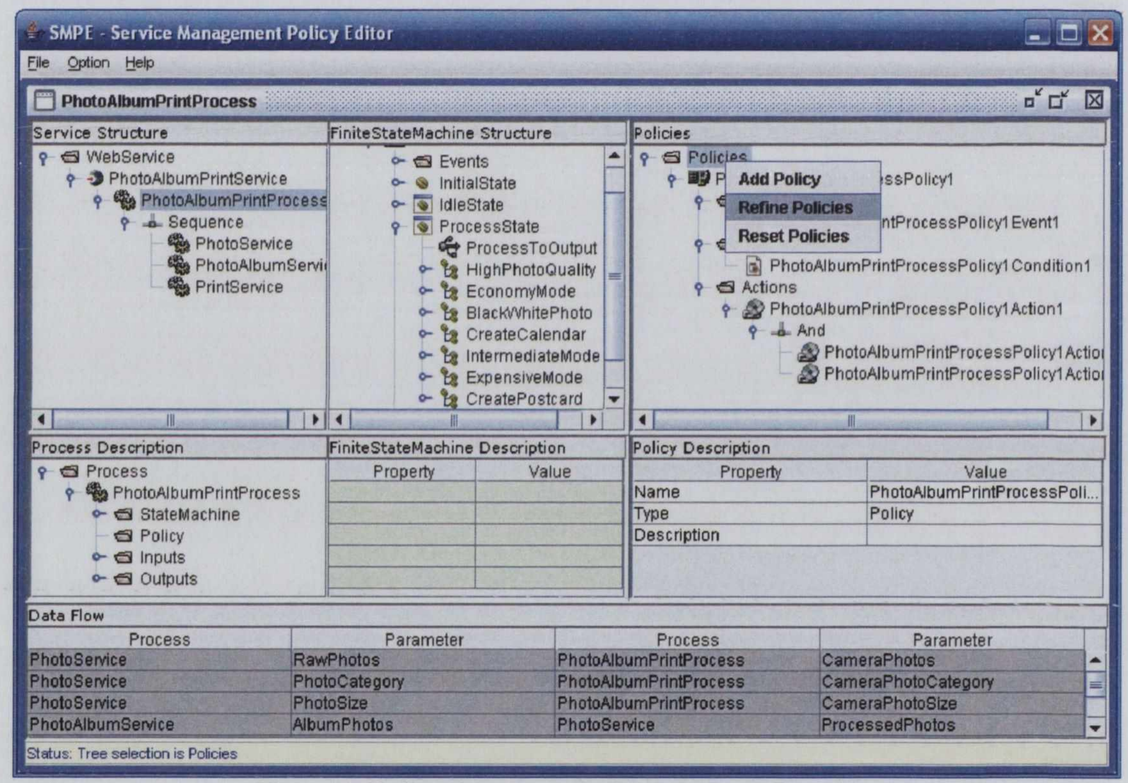
ii. Verify that there are no errors from the policy refinement report produced;

iii. Check to see the generated enforceable policies;

iv. Save the policies created for this service;

Using the same composite service PhotoAlbumPrintService and the same specified management policy as in the previous section (see section 4.7.1).

i. Management policies for this service can be refined by right clicking on Policies and select Refine Policies from the dropdown menu (see Figure 4-42).



Policy Description

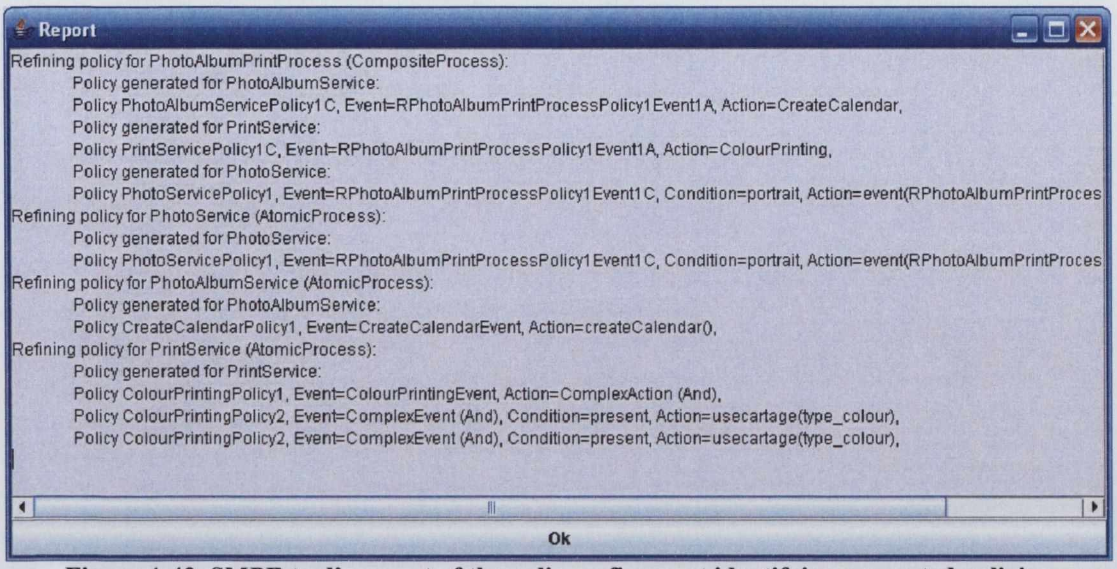
Property	Value
Name	PhotoAlbumPrintProcessPoli...
Type	Policy
Description	

Data Flow

Process	Parameter	Process	Parameter
PhotoService	RawPhotos	PhotoAlbumPrintProcess	CameraPhotos
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	CameraPhotoCategory
PhotoService	PhotoSize	PhotoAlbumPrintProcess	CameraPhotoSize
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos

Figure 4-42, Starting the automated policy refinement

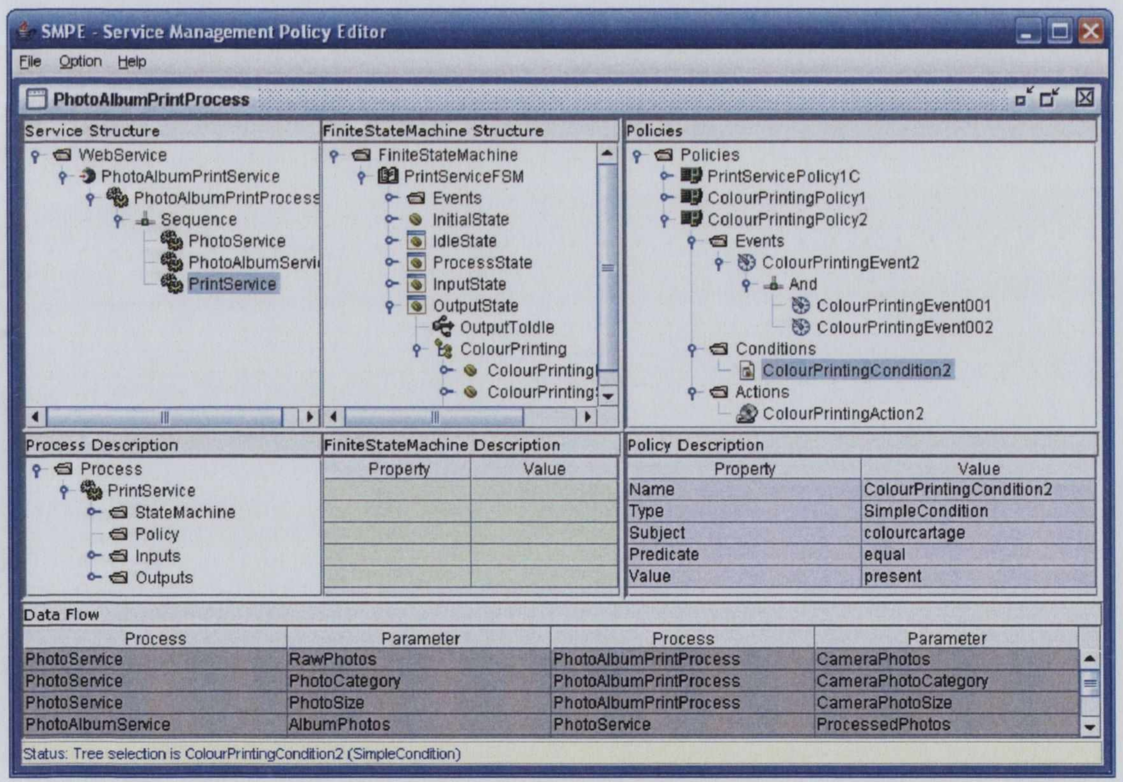
ii. The SMPE tool notifies the user with a report of the success of the automated policy refinement process. In this case the tool reports that there were no errors and it provides a list of enforceable policies generated (see Figure 4-43).



```
Refining policy for PhotoAlbumPrintProcess (CompositeProcess):
        Policy generated for PhotoAlbumService:
        Policy PhotoAlbumServicePolicy1C, Event=RPhotoAlbumPrintProcessPolicy1Event1A, Action=CreateCalendar,
        Policy generated for PrintService:
        Policy PrintServicePolicy1C, Event=RPhotoAlbumPrintProcessPolicy1Event1A, Action=ColourPrinting,
        Policy generated for PhotoService:
        Policy PhotoServicePolicy1, Event=RPhotoAlbumPrintProcessPolicy1Event1C, Condition=portrait, Action=event(RPhotoAlbumPrintProces
Refining policy for PhotoService (AtomicProcess):
        Policy generated for PhotoService:
        Policy PhotoServicePolicy1, Event=RPhotoAlbumPrintProcessPolicy1Event1C, Condition=portrait, Action=event(RPhotoAlbumPrintProces
Refining policy for PhotoAlbumService (AtomicProcess):
        Policy generated for PhotoAlbumService:
        Policy CreateCalendarPolicy1, Event=CreateCalendarEvent, Action=createCalendar(),
Refining policy for PrintService (AtomicProcess):
        Policy generated for PrintService:
        Policy ColourPrintingPolicy1, Event=ColourPrintingEvent, Action=ComplexAction (And),
        Policy ColourPrintingPolicy2, Event=ComplexEvent (And), Condition=present, Action=usecartage(type_colour),
        Policy ColourPrintingPolicy2, Event=ComplexEvent (And), Condition=present, Action=usecartage(type_colour),
```

Figure 4-43, SMPE tool's report of the policy refinement identifying generated policies

iii. Details of these enforceable policies can be seen by browsing the relevant services from the top left panel, and its policies are loaded in the top right panel, where their aspects are loaded in the bottom right panel when selected (see Figure 4-44).



Property	Value
Name	ColourPrintingCondition2
Type	SimpleCondition
Subject	colourcartage
Predicate	equal
Value	present

Process	Parameter	Process	Parameter
PhotoService	RawPhotos	PhotoAlbumPrintProcess	CameraPhotos
PhotoService	PhotoCategory	PhotoAlbumPrintProcess	CameraPhotoCategory
PhotoService	PhotoSize	PhotoAlbumPrintProcess	CameraPhotoSize
PhotoAlbumService	AlbumPhotos	PhotoService	ProcessedPhotos

Status: Tree selection is ColourPrintingCondition2 (SimpleCondition)

Figure 4-44, SMPE tool displaying the generated refined policies for the constituent services

iv. The refinement of this high-level management policy for the composite web service PhotoAlbumPrintService is now complete and can be saved. This is achieved by clicking File, then Save. When saving, generated enforceable policies are saved on a separate file, which prevents from getting mixed with high-level policies and allows the file to be directly consumed by a policy engine.

This section demonstrates how the SMPE tool can be used to accomplish the Management Policy Refinement Activity and the Management Policy Generation Activity of the MAWS methodology by automatically generating refined enforceable policies to manage adaptive (composite) services.

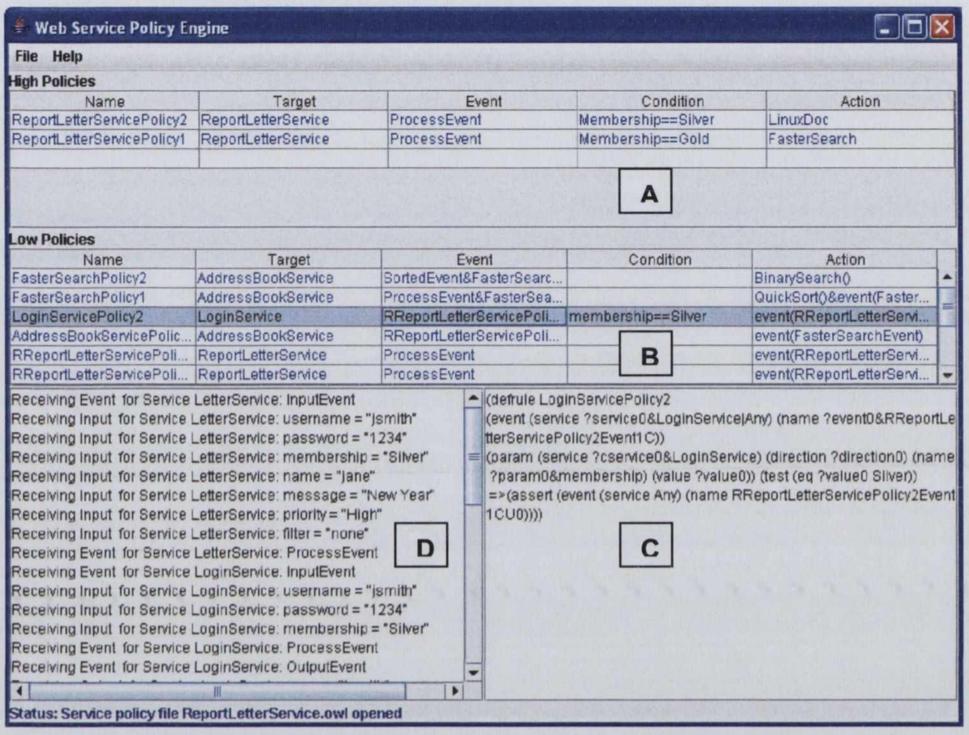
4.8 Adaptive Web Service PBMS Evaluation Platform

In order to verify that the generated enforceable policies can manage the adaptive composite services according to the specified high-level management policies, a policy evaluation framework for web services was needed. The policy evaluation platform was implemented to validate case studies during runtime. It offers the necessary support to execute adaptive web services at runtime and provide runtime traces for analysis.

The policy evaluation framework can host adaptive web services, and together with their enforceable policies for managing their adaptive behaviours. The evaluation policy platform consists of a web service container for executing the adaptive web services, and a policy engine, which consumes the generated low-level policies and enforces them upon the hosted web services for managing their adaptive behaviours during their execution.

Apache Tomcat [48] and Apache Axis2 [49] were used as the web service container to host these adaptive web services. Axis2 has Java2Wsdl and Wsdl2Java which allows for the generation of WSDL for web services implemented in Java or for the generation of web service skeleton code in Java for WSDL description of web services. Both Apache Tomcat and Apache Axis2 are very widely used as web service containers in industry and academia and therefore provide an authentic implementation environment for adaptive web services.

This framework also includes a policy engine, which was build upon the Jess rule engine [47]. The policy engine consumes generated low-level policies expressed in OWL and transforms them into Jess rules. These enforceable policies, expressed as Jess rules, are loaded into the Jess rule engine to manage the relevant adaptive web services in the web service container through a Remote Method Invocation (RMI) connection. This RMI connection ensures that the policy engine is informed of events from the adaptive web services and that actions assigned by a triggering policy are invoked on the relevant adaptive web service. Thus, allowing several adaptive web services to be enforced by a single policy engine.



High Policies

Name	Target	Event	Condition	Action
ReportLetterServicePolicy2	ReportLetterService	ProcessEvent	Membership==Silver	LinuxDoc
ReportLetterServicePolicy1	ReportLetterService	ProcessEvent	Membership==Gold	FasterSearch

Low Policies

Name	Target	Event	Condition	Action
FasterSearchPolicy2	AddressBookService	SortedEvent&FasterSearc...		BinarySearch()
FasterSearchPolicy1	AddressBookService	ProcessEvent&FasterSea...		QuickSort()&event(Faster...
LoginServicePolicy2	LoginService	RReportLetterServicePoli...	membership==Silver	event(RReportLetterServi...
AddressBookServicePolic...	AddressBookService	RReportLetterServicePoli...		event(FasterSearchEvent)
RReportLetterServicePoli...	ReportLetterService	ProcessEvent		event(RReportLetterServi...
RReportLetterServicePoli...	ReportLetterService	ProcessEvent		event(RReportLetterServi...

```
Receiving Event for Service LetterService: InputEvent
Receiving Input for Service LetterService: username = "jsmith"
Receiving Input for Service LetterService: password = "1234"
Receiving Input for Service LetterService: membership = "Silver"
Receiving Input for Service LetterService: name = "Jane"
Receiving Input for Service LetterService: message = "New Year"
Receiving Input for Service LetterService: priority = "High"
Receiving Input for Service LetterService: filter = "none"
Receiving Event for Service LetterService: ProcessEvent
Receiving Event for Service LoginService: InputEvent
Receiving Input for Service LoginService: username = "jsmith"
Receiving Input for Service LoginService: password = "1234"
Receiving Input for Service LoginService: membership = "Silver"
Receiving Event for Service LoginService: ProcessEvent
Receiving Event for Service LoginService: OutputEvent
```

```
(defrule LoginServicePolicy2
(event (service ?service0&LoginService|Any) (name ?event0&RReportLe
tterServicePolicy2Event1C))
(param (service ?cservice0&LoginService) (direction ?direction0) (name
 ?param0&membership) (value ?value0)) (test (eq ?value0 Silver))
 =>(assert (event (service Any) (name RReportLetterServicePolicy2Event
1CU0))))
```

Status: Service policy file ReportLetterService.owl opened

Figure 4-45, Snapshot of the Policy Evaluation Framework

The policy engine was implemented with a GUI to facilitate observing the policy execution, as depicted in Figure 4-45. Its top panel (A) shows the high-level management policy, while the middle panel (B) shows the refined enforceable policies. The bottom right panel (C) shows the Jess rule for the selected low-level enforceable policy. And the bottom left panel (D) shows a running trace of the policy engine.

In order to ensure that the policy evaluation platform is able to manage these adaptive web services, the following steps are suggested for web service developers in their implementation, and were used in the implementation of the adaptive web services for the case studies described in section 5.2:

1. Create a project in Eclipse for a web service;
2. Create a WSDL file for a web service;
3. Describe this web service in the WSDL file with a single request

4. Use Wsdl2Java tool with the options –ss –sd –uri <wsdl file>, this will generate web service skeleton java files;
5. Edit build.xml and add PolicyEngineProxy.jar to classpath
6. Edit service.xml and add logging module to this service
7. Edit the web service skeleton java file and add the following lines:
   a. Add PolicyEngineProxy attribute to this service and initialize it in the service's constructor with the parameter this;
   b. Add line *engine.event(servicename, InputEvent)* in the service main function before reading the service's input parameters;
   c. Add line *engine.event(servicename, ProcessEvent)* in the service main function after reading the service's input parameters;
   d. Add line *engine.event(servicename, OutputEvent)* in the service main function before writing the service's output parameters;
8. Finally, add the desired adaptive behaviours as public functions.

A tool called SoapUI [45] was used to invoke the web services. This tool facilitates the creation of soap requests and allows users to invoke the soap request with different parameters. Service responses are then displayed to the user through the SoapUI's GUI.

4.9 Summary

This chapter illustrated how the integrated set of tools was implemented to support the MAWS methodology in describing web services and their adaptive behaviours using a combination of OWL-S and FSM. In addition these integrated tools also support the specification of management policies and automated refinement of these policies, generating enforceable policies to manage adaptive web services. The design and implementation of each of these integrated tools was explained in detail and included how the respective enabling technologies were used to achieve the functionality of each of their main components. Lastly, a demonstration of the use of these tools is provided together with screenshots.

In the next chapter, Chapter 5, a description of how these tools operate with different case studies and how they undergo a usability test is presented. Furthermore, the results of each of their experiments and conclusions are also presented.

5 Evaluation

5.1 Introduction

In order to evaluate the MAWS approach, three tools were developed. This chapter evaluates these innovative tools as proof of concept, in particular the SABE and SMPE tools. These prototype tools, as proofs of concept, have undergone experiments to evaluate their usability and functionality. More specifically these experiments focus on validating the approach these tools support against the objectives set out in this thesis. The first experiment is a functional evaluation with case studies used to validate the approach proposed using the implemented integrated tools. Two application areas for adaptive web services were chosen, namely personalisation and business context adaptation. Each case study validates the functionality of the tools in creating discrete policies to manage adaptive composite services under different circumstances.

Usability experiments were conducted for the prototype tools SABE and SMPE to appraise their usability by measuring the degree of ease with which users can complete specific tasks using these tools. Due to time limitations and a limited number of users, usability experiments were conducted only for the SABE and SMPE, and not for the WSDE tool, since it is not novel to the field of describing web services as OWL-S.

Finally, a discussion of related work to the approach proposed in this thesis is presented at the end of this chapter. This discussion compares other approaches in the domain of adaptive web services and policy refinement to the proposed approach, depicting any relevance and contrasting any differences.

5.2 Functionality Evaluation

The aim of the functionality evaluation is to evaluate SABE and SMPE tools under different scenarios but more importantly, scenarios that provide adaptive web services which demonstrate higher level of complexity than the adaptive web services used in the usability experiments. Two scenarios were designed in different application areas to evaluate the functionality of these tools, and to validate their flexibility as a set of integrated tools for managing adaptive composite services. The adaptive services chosen for these case studies were selected to demonstrate how the proposed approach can successfully adapt web services in the areas of personalisation, context changes, and business model flexibility. Therefore in order to evaluate the MAWS methodology and its supporting tools, these two case studies had to be defined and executed using the policy evaluation platform described in section 4.8.

For each case study, adaptive composite services were implemented in Java, the processes in the MAWS methodology were followed, and its supporting tools were used: (i) the WSDE tool was used for describing the atomic web services semantically, (ii) the SABE tool was used for describing their adaptive behaviours, (iii) the WSDE tool was used for describing the composition of composite web services; (iv) the SABE tool was used for aggregating the adaptive behaviours of composite services; (v) the SMPE tool was used for specifying management policies and for automatically refining them into low-level enforceable policies. Finally, the adaptive web services and their enforceable policies were loaded onto the policy evaluation platform[7], and runtime traces were recorded. The policy evaluation platform offers the necessary runtime support for the following case studies to be evaluated.

5.2.1 Case Study – Personalised Holiday Service

This case study demonstrates how the integrated tools can be used to describe and manage adaptive behaviours of a composite web service called Personal Holiday service. This web service allows a user to login and book a holiday trip, which includes a return flight to an exotic destination of their choice and accommodation in a resort hotel, and finally allows them to pay for the trip through a payment service. This adaptive composite service is depicted in Figure 5-1.

The Personal Holiday service is more complex than the composite web service used in the usability experiments; for instance, this composite service is itself composed of another composite service. The multi-level hierarchy of constituent services allows this case study to demonstrate how the integrated tools can automatically aggregate the adaptive behaviours of this web service in a reversed

[7] Implementation of the policy evaluation platform is described in section 4.8

hierarchical manner. In addition, it shows that the integrated tools can automatically refine the specified policies, generating enforceable policies for this web service in a hierarchical manner.

Furthermore, this case study provides an application scenario that demonstrates how the proposed approach to adaptive web services can empower users using these web services. It shows the flexibility of integrating (individual and group) user preferences into web services. It shows how adaptive web services can be personalised to suit user's needs or preferences, as they are specified through policies. A description of this adaptive composite service together with the specified high-level management policies and their enforceable policy counterparts is presented.

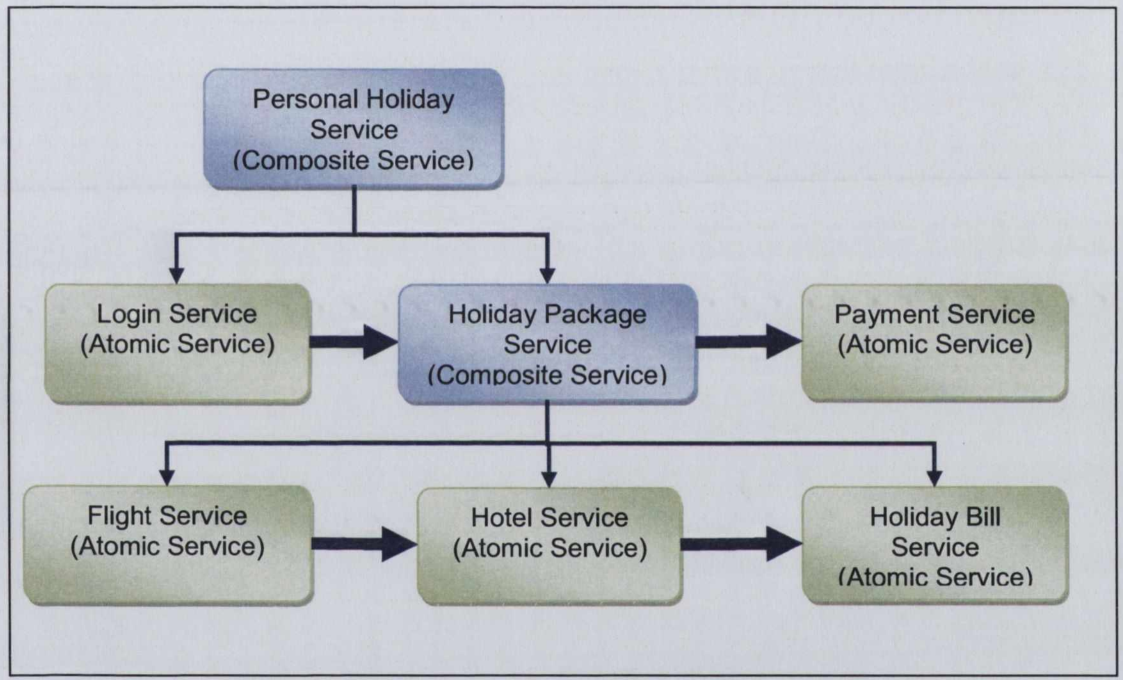

Personal Holiday Service (Composite Service)
Login Service (Atomic Service)
Holiday Package Service (Composite Service)
Payment Service (Atomic Service)
Flight Service (Atomic Service)
Hotel Service (Atomic Service)
Holiday Bill Service (Atomic Service)

Figure 5-1, Diagram of the Personalised Holiday Service

Personal Holiday Service

The Personal Holiday Service, depicted in Figure 5-1, is composed of:

- a login service, for authenticating the user's credentials;
- a holiday package service; which provides the user with a holiday package including a return flight and hotel, and is composed of:
    - a flight service, for booking a return flight to a resort destination;
    - a hotel service, for reserving a room in one of the hotels from the resort destination;

    - a holiday bill service, for generating the holiday bill by adding the total cost;
- a payment service, which charges the user with the total amount for the holiday trip to his/her visa card;

However, imagine that it was also possible to personalise this service to meet your preferences or needs, such as if you were on a business trip, vegetarian or in need of wheelchair accessibility. In order to accommodate the personalisation of these services, adaptive behaviours were added to these services together with their semantic descriptions.

Flight Service's Adaptive Behaviours

The flight service was improved with the following adaptive behaviours:

- Accessibility, changes the service's behaviour to allocate a seat beside the door for accessibility;
- Vegetarian, changes the service's behaviour to serve the passenger with a vegetarian meal;
- BusinessClass, changes the service's behaviour to allocate a business class seat on the flight;
- HighClass, changes the service's behaviour to allocate a first class seat on the flight;

Hotel Service's Adaptive Behaviour

The hotel service was implemented with the following adaptive behaviours:

- Accessibility, changes the service's behaviour to allocate a room with wheelchair accessibility;
- Vegetarian, changes the service's behaviour to prepare vegetarian menu for the guest;
- BusinessClass, changes the service's behaviour to prepare the reserved room for business purposes (desk, internet connection, conference phone);
- HighClass, changes the service's behaviour to reserve the presidential suite;

Payment Service's Adaptive Behaviour

The payment service has the default behaviour of charging users on a visa card. This service was improved with the following adaptive behaviours:

- MasterCard, changes the service's behaviour to charge users on his/her master card;
- AmericanExpress, changes the service's behaviour to charge users on his/her American Express card;
- TravelVoucher, changes the service's behaviour to utilise travel vouchers to pay for the trip;

Management Policies

In order to demonstrate how this adaptive composite service can be personalised for a particular user or a group of users, two set of policies were created, one for each:

- Susan Smith is someone in a wheelchair, who likes to go on holidays. However, there needs to be some accessibility for her, both in the flight as well as in the hotel. Thus, the policy specified for her is as follows:
  - *"ON Process Event, IF username = Susan.Smith AND Membership = Frequent Flyer, THEN Action: Accessibility".*
- John Murphy is a member of the sales team in a multinational company. John and his team are in charge of advertisement in different countries, and while in those countries they need to present their new products. Therefore, in order to prepare for their presentations, they need a place to do their job, i.e. both the flight and accommodation need to be business class. Thus, the policy specified for him and his team members is as follows:
  - *"ON Process Event, IF Membership = SalesGroup, THEN Action: BusinessClass;*

The high-level policy specified for John Murphy and her team using the SMPE tool is shown in Figure 5-2, described as Obligation Policy Ontology Model.

```
<policy:Policy rdf:ID="PersonalHolidayServicePolicy2">
<rdfs:comment></rdfs:comment>
<policy:target rdf:resource="PersonalHolidayProcess.owl#
PersonalHolidayService"/><policy:event rdf:resource="
#PersonalHolidayServicePolicy2Event1"/><policy:condition
rdf:resource=" #PersonalHolidayServicePolicy2Condition1"/>
<policy:action rdf:resource="
#PersonalHolidayServicePolicy2Action1"/></policy:Policy>
<policy:SimpleEvent rdf:ID="PersonalHolidayServicePolicy2Event1">
<policy:value>ProcessEvent</policy:value></policy:SimpleEvent>
<policy:SimpleCondition rdf:ID="
PersonalHolidayServicePolicy2Condition01"><policy:subject>
<policy:Subject rdf:ID="Membership"/></policy:subject>
<policy:predicate><policy:Predicate rdf:ID="equal"/>
</policy:predicate><policy:value> SalesGroup</policy:value>
</policy:SimpleCondition><policy:SimpleAction
rdf:ID="PersonalHolidayServicePolicy2Action1">
<policy:value>BusinessClass</policy:value></policy:SimpleAction>
```

Figure 5-2, A snippet of the second management policy specified for the Personal Holiday Service

Next, the SMPE tool was used to automatically refine these management policies into enforceable policies. The tables below show a summary of the enforceable policies generated for the Personal Holiday Service and its constituent services.

PersonalHolidayService (CompositeProcess)			
Name	**Event**	**Condition**	**Action**
Policy1	ProcessEvent		event(RPersonalHolidayServicePolicy1Event1C0)
Policy2	ProcessEvent		event(RPersonalHolidayServicePolicy2Event1C0)
Policy1CU	RPersonalHolidayService Policy1Event1CU0		event(RPersonalHolidayServicePolicy1Event1aA)
Policy2CU	RPersonalHolidayService Policy2Event1CU0		event(RPersonalHolidayServicePolicy2Event1aA)

Table 4, Enforceable policies generated for Personal Holiday Service

LoginService (AtomicProcess)			
Name	**Event**	**Condition**	**Action**
Policy1	RPersonalHolidayService Policy1Event1C0	(name==Susan.Smith)&& (member==FrequentFlyer)	event(RPersonalHolidayServicePolicy 1Event1CU0)
Policy2	RPersonalHolidayService Policy2Event1C0	member==SalesGroup	event(RPersonalHolidayServicePolicy 2Event1CU0)

Table 5, Enforceable policies generated for Login Service

HolidayPackageService (CompositeProcess)			
Name	**Event**	**Condition**	**Action**
Policy1C	RPersonalHolidayService Policy1Event1aA		event(HolidayPackageServicePolicy1CEvent2aA)&& event(HolidayPackageServicePolicy1CEvent2bA)
Policy2C	RPersonalHolidayService Policy2Event1aA		event(HolidayPackageServicePolicy2CEvent2aA)&& event(HolidayPackageServicePolicy2CEvent2bA)

Table 6, Enforceable policies generated for Holiday Package Service

FlightService (AtomicProcess)			
Name	**Event**	**Condition**	**Action**
Policy1CC	HolidayPackageServicePolicy1CEvent2aA		event(AccessibilityEvent)
AccessibilityPolicy1	ProcessEvent&& AccessibilityEvent		Accessibility()
Policy2CC	HolidayPackageServicePolicy2CEvent2aA		event(BusinessClassEvent)
BusinessClassPolicy1	ProcessEvent&& BusinessClassEvent		BusinessClass()

Table 7, Enforceable policies generated for Flight Service

HotelService (AtomicProcess)			
Name	**Event**	**Condition**	**Action**
Policy1CC	HolidayPackageServicePolicy1CEvent2bA		event(AccessibilityEvent)
AccessibilityPolicy1	ProcessEvent&& AccessibilityEvent		Accessibility()
Policy2CC	HolidayPackageServicePolicy2CEvent2bA		event(BusinessClassEvent)
BusinessClassPolicy1	ProcessEvent&& BusinessClassEvent		BusinessSuite()

Table 8, Enforceable policies generated for Hotel Service

In order to validate these policies, the adaptive web services described above were implemented as shell web services, and they were loaded into the policy evaluation framework to evaluate these policies. The enforceable policies were translated from their ontology model to jess rules by the policy evaluation framework before being executed to manage these adaptive web services. Figure 5-3 shows the generated low-level policies as jess rules for managing the Personal Holiday Service.

```
(defrule HotelServicePolicy2CC
(event (service ?service0&HotelService|Any) (type ?type0) (name
?event0&HolidayPackageServicePolicy2CEvent2A))
=>(assert (event (service Any) (type Policy) (name BusinessClassEvent)))
(assert (action (service ?service0) (name ?event0))))

(defrule LoginServicePolicy2
(event (service ?service0&LoginService|Any) (type ?type0) (name
?event0&RPersonalHolidayServicePolicy2Event1C)) (event (service
?cservice0) (type ?ctype0) (name ?cevent0&~IdleEvent))
(param (service ?cservice0&LoginService) (direction ?direction0) (name
?param0&member) (value ?value0)) (test (eq ?value0 Marketing))
=>(assert (event (service Any) (type Policy) (name
RPersonalHolidayServicePolicy2Event1CU0))) (assert (action (service
?service0) (name ?event0))))

(defrule RPersonalHolidayServicePolicy2CU
(event (service ?service0&PersonalHolidayService|Any) (type ?type0) (name
?event0&RPersonalHolidayServicePolicy2Event1CU0))
=>(assert (event (service Any) (type Policy) (name
RPersonalHolidayServicePolicy2Event1A))) (assert (action (service
?service0) (name ?event0))))

(defrule HolidayPackageServicePolicy2C
(event (service ?service0&HolidayPackageService|Any) (type ?type0) (name
?event0&RPersonalHolidayServicePolicy2Event1A))
=>(assert (event (service Any) (type Policy) (name
HolidayPackageServicePolicy2CEvent2A))) (assert (action (service
?service0) (name ?event0))))

(defrule FlightServicePolicy2CC
(event (service ?service0&FlightService|Any) (type ?type0) (name
?event0&HolidayPackageServicePolicy2CEvent2A))
=>(assert (event (service Any) (type Policy) (name BusinessClassEvent)))
(assert (action (service ?service0) (name ?event0))))

(defrule RPersonalHolidayServicePolicy2
(event (service ?service0&PersonalHolidayService|Any) (type ?type0) (name
?event0&ProcessEvent))
=>(assert (event (service Any) (type Policy) (name
RPersonalHolidayServicePolicy2Event1C))))
```

Figure 5-3, A snippet of the enforceable policies as Jess Rules for the second management policy

The request and response made by Susan Smith to the Personal Holiday Service can be seen in Table 9.

Request		**Response**	
Parameter	**Value**	**Parameter**	**Value**
Username:	*Susan.Smith*	Approved	*true*
Password:	*1234*	Flight class	*EconomyClass*
Member:	*FrequentFlyer*	Flight number	*FR235*
Origin:	*New York*	Itinerary	*NYPR_Hilton_0204*
Destination:	*Paris*	Hotel name	*Hilton*
Going Date:	*01/03/2007*	Hotel address	*Charles De Gauld*
Leaving Date:	*05/03/2007*	Hotel reservation	*Hilton_02_4*
Person :	*1*	Hotel stars	*5*

		Hotel facilities	*Swimming Pool Game Room Sauna*
		Brochure	*English, French*
		Total	*2530.0*

Table 9, Susan Smith's request and response to the Personal Holiday Service

The request made by John Murphy to the same instance of the Personal Holiday Service and the response returned can be seen in Table 10.

Request		**Response**	
Parameter	**Value**	**Parameter**	**Value**
Username:	*John.Murphy*	Approved	*true*
Password:	*1234*	Flight class	*EconomyClass*
Member:	*SalesGroup*	Flight number	*BA655*
Origin:	*Toronto*	Itinerary	*TOLO_Mariott_0102*
Destination:	*London*	Hotel name	*Mariott*
Going Date:	*10/04/2007*	Hotel address	*Abbey Street*
Leaving Date:	*12/04/2007*	Hotel reservation	*Mariott_01_2*
Person :	*1*	Hotel stars	*4*
		Hotel facilities	*Swimming Pool Game Room*
		Brochure	*English*
		Total	*1650.0*

Table 10, John Murphy's request and response to the Personal Holiday Service

Runtime traces were recorded with the use of the policy evaluation framework for these two requests made to a single instance of the Personal Holiday Service; one made by Susan Smith and the other made by John Murphy. Figure 5-4 shows the content of a runtime log file for the first request made by user Susan Smith and Figure 5-5 show the trace log of the same service instance at runtime for the second request made by user John Murphy from the sales team. Please note the difference in the trace logs where changes in the service behaviour are indicated with $$.

```
[Login] Reading user list from file
[Login] username = Susan.Smith
[Login] password = 1234
[Login] membership = FrequentFlyer
[Login] Susan Smith has successfully login
[Flight] Reading user list from file
[Flight] Reading airline list from file
[Flight] Processing flight reservation request
[Flight] Searching for a flight from New York to Paris
[Flight] $$Reserving flight seat with Wheelchair access
[Flight] A flight was found for 300.0 euro
[Hotel] Reading user list from file
[Hotel] Reading hotel list from file
[Hotel] Processing hotel reservation request
[Hotel] Searching for a hotel in Paris
[Hotel] Hotel found for the desired holiday location Hilton
[Hotel] $$Reserving room with Wheelchair access
[HotelBill] Processing Holiday Bill
[HotelBill] Holiday package adds up to 2300.0
[HotelBill] Sales tax is 10.0
[Payment] Reading user list from file
[Payment] Processing Payment request for a bill of 2300.0
[Payment] Retrieving Visa Card Number
[Payment] Retrieving Visa Expiry Date
[Payment] Validating Visa Card
[Payment] Charging the user Susan Smith: 2530.0
[Payment] Payment of 2530.0 including sales tax of 10.0% has been
approved
```

Figure 5-4, Runtime trace for Susan Smith using Personal Holiday Service

```
[Login] username = John.Murphy
[Login] password = 1234
[Login] membership = SalesGroup
[Login] Sending Event ProcessEvent
[Login] John Murphy has successfully login
[Flight] Reading user list from file
[Flight] Reading airline list from file
[Flight] Processing flight reservation request
[Flight] Searching for a flight from Toronto to London
[Flight] $$Reserving Business class flight
[Flight] A flight was found for 700.0 euro
[Hotel] Reading user list from file
[Hotel] Reading hotel list from file
[Hotel] Processing hotel reservation request
[Hotel] Searching for a hotel in London
[Hotel] Hotel found for the desired holiday location Mariott
[Hotel] $$Reserving Business suite
[HotelBill] Processing Holiday Bill
[HotelBill] Holiday package adds up to 1500.0
[HotelBill] Sales tax is 10.0
[HotelBill] Reading user list from file
[Payment] Reading user list from file
[Payment] Processing Payment request for a bill of 1500.0
[Payment] Retrieving Visa Card Number
[Payment] Retrieving Visa Expiry Date
[Payment] Validating Visa Card
[Payment] Charging the user John Murphy: 1650.0
[Payment] Payment of 1650.0 including sales tax of 10.0% has been
```

Figure 5-5, Runtime trace for John Murphy using Personal Holiday Service

It can be noted that the same web service instance behaved differently for these two requests. In other words, this web service was personalised for these two users according to their policies (**$$** indicates the changes in the service behaviour). For full detail of the artefacts used in this case study – see Appendix F.

This case study illustrated how the integrated tools successfully described user centric adaptive behaviours of a composite web service. Furthermore, it illustrated a successful automated policy refinement of management policies specified to personalise this web service to suit the preference of two different users. This is just a small sample of how web services can be personalised through adaptive behaviours using this approach.

One might argue that this personalisation can be implemented within the web services. And indeed they can as shown in the In-house approach (section 2.3.1). But the key to the novel approach is to have the intelligence expressed externally through policies that allows web services to be dynamically adaptive in a declarative manner. Furthermore, the use of policies allows the changes in the personalisation without the need to recode web services.

The case study showed how adaptive behaviours of the constituent web services were aggregated together, using the SABE tool, for this composite web service. And also, that this aggregation was performed in a hierarchical manner over different levels of composition. More importantly, it showed how the SMPE tool successfully refined the specified management policies, in an automated manner, into low-level policies for the relevant constituent adaptive atomic web services.

This case study also demonstrated how the SMPE tool was capable of refining complex high-level policies. It was able to automatically refine both policies with multiple conditions and with multiple actions. This refinement also includes mapping policies that are created for the different web services in the composition path linking the high-level policies to the low-level policies. The policy refinement process is also done in a hierarchical manner, so it can be performed over several levels of composition. This was demonstrated through this case study, which contains multiple levels of composition.

5.2.2 Case Study – Notification Service

This case study demonstrates how the integrated tools can be used to describe and manage adaptive behaviours of a composite service called Notification service. When the Notification Service receives a notification request, it formats the notification's message, finds the address of the recipient, and contacts the recipient by phone and email with the notification's message. This web service, depicted in Figure 5-6, also has a composition more complex than the composite service used in the usability experiment with more number of constituent services and a multi-level hierarchy of constituent services.

Hence, this case study demonstrated how the integrated tools can automatically aggregate the adaptive behaviours of this web service in a reversed hierarchical manner. In addition, it showed that the integrated tools can automatically refine the specified policies, generating enforceable policies for this web service in a hierarchical manner.

However, this case study differs from the previous one since this case study focus on demonstrating how adaptive web services can be described and managed by policies in order to make their behaviour dynamic to changes in context and to changes in business logic. To demonstrate how the Notification service could change its behaviours dynamically for changes both in context and in business logic, a larger set of policies were specified than previous case study. By using the integrated tools to refine these policies, a large number of enforceable policies were generated targeting different constituent services. Furthermore, these enforceable policies changed the web services behaviour appropriately and did not conflict with each other, as seen through the runtime trace logs.

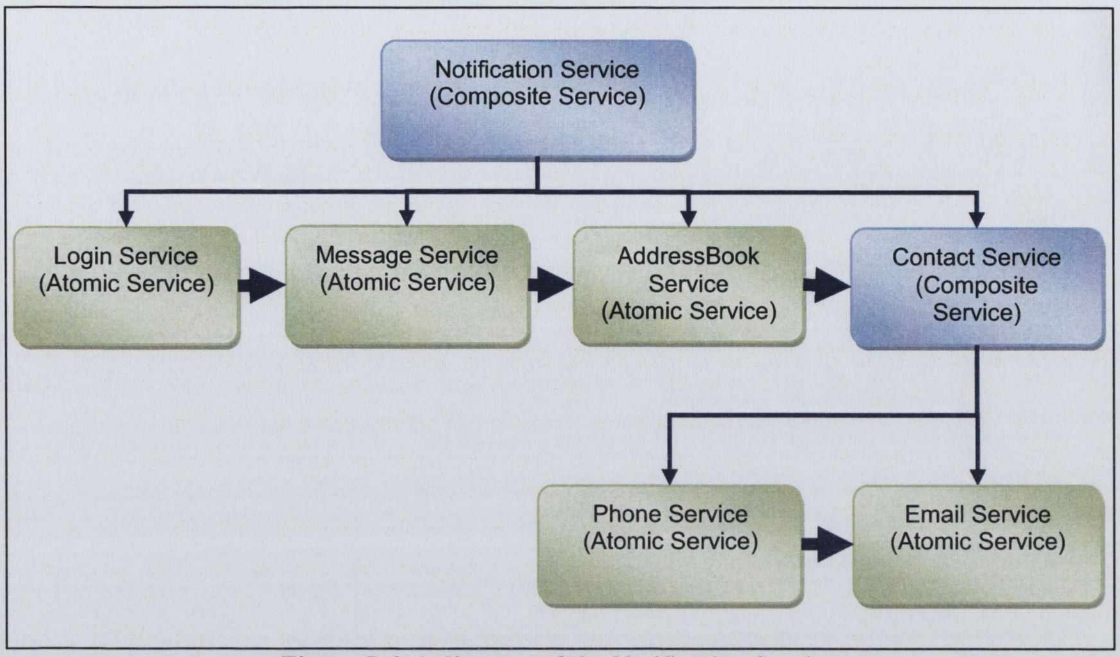


Figure 5-6, A diagram of the Notification Service

Notification Service's Adaptive Behaviours

The Notification service is composed of:

- a login service, for authenticating the user's credentials;
- a message service, which formats the message content for the recipient to be notified with;
- an address book service, which finds the address (email and phone number) of the recipient to be contacted;
- a contact service; which contacts the recipient with the message provided, and composed of:
  - a phone service, which notifies the recipient by phone with the given message;
  - an email service, which notifies the recipient by email with the given message;

In order to accommodate changes in business logic or environment context, this service was composed of adaptive web services.

Message Service's Adaptive Behaviours

The message service was implemented with the following adaptive behaviours:

- MessageInSpanish, which translates the notification message to Spanish;

- MessageInFrench, which translates the notification message to French;
- ShortMessageMode, truncates the notification message to 120 characters;

Address Book Service's Adaptive Behaviours

The Address Book service was implemented with the following adaptive behaviours:

- QuickSearch, changes the behaviour of this service to perform a binary search to a list sorted using QuickSort algorithm;
- SimpleSearch, changes the behaviour of this service to perform a binary search to a list sorted using a simple progression algorithm;
- EmergencyContact, modifies the service to return high priority contact details;

Phone Service's Adaptive Behaviours

The Phone service was implemented with the following adaptive behaviours:

- Authentication, requests that the recipient authenticates using a pin;
- Tenacious, modifies the service to retry multiple times if recipient is not reached;

Email Service's Adaptive Behaviours

The Email service was implemented with the following adaptive behaviours:

- RichContent, transform the email's content to use HTML format;
- Encryption, encrypts the email's content;
- Compression, compresses the email's content before sending it;

Management Policies

Many sets of management policies can be created to manage these adaptive web services to accommodate different scenarios of context changes or changes in business logic. Three sets of management policies were specified to demonstrate how this adaptive composite service could be dynamically adaptive to context change and to changes in business logic:

- Use optimised algorithm and high-quality content for gold subscribers, and save on resources and less efficient algorithm for bronze subscribers (while the service's default behaviour would be for silver subscribers):
    - *"IF membership = Gold, THEN Action: QuickSearch AND RichContent"*
    - *"IF membership = Bronze, THEN Action: SimpleSearch AND ShortMessageMode"*
- Use emergency contact and make sure it contacts the right person in case of a high-priority notification:
    - *"IF priority = high, THEN Action: EmergencyContact AND Authentication"*
- Translate the notification message to French for users of the Club de Leon:
    - *"IF membership = LeonClub, THEN Action: MessageInFrench"*

The lexical form of the high-level management policy specified for high-priority notifications (third management policy), using the SMPE tool, is shown in Figure 5-7.

```
<policy:Policy rdf:ID="NotificationServicePolicy3">
<rdfs:comment></rdfs:comment>
<policy:target rdf:resource="NotificationProcess.owl#
NotificationService"/><policy:event><policy:SimpleEvent rdf:ID="
NotificationServicePolicy3Event1"><policy:value>ProcessEvent
</policy:value></policy:SimpleEvent></policy:event>
<policy:condition><policy:SimpleCondition rdf:ID="
NotificationServicePolicy3Condition1"><policy:subject>
<policy:Subject rdf:ID="Priority"/></policy:subject>
<policy:predicate rdf:resource="#equal"/>
<policy:value>high</policy:value>
</policy:SimpleCondition></policy:condition><policy:action>
<policy:ComplexAction rdf:ID="NotificationServicePolicy3Action1">
<rdfs:first><policy:AndList><rdfs:first><policy:SimpleAction
rdf:ID="NotificationServicePolicy3Action101">
<policy:value>EmergencyContact</policy:value>
</policy:SimpleAction></rdfs:first><rdfs:rest>
<policy:SimpleAction rdf:ID="
NotificationServicePolicy3Action102">
<policy:value>Authentication</policy:value></policy:SimpleAction>
</rdfs:rest></policy:AndList></rdfs:first><rdfs:rest></rdfs:rest>
</policy:ComplexAction></policy:action></policy:Policy>
```

Figure 5-7, Snippet of the third management policy specified for the Notification Service

Next, the SMPE tool was used to automatically refine these management policies into enforceable policies. Tables below show a summary of the enforceable policies generated for the Notification Service and its constituent services.

NotificationService (CompositeProcess)

Name	Event	Condition	Action
Policy1	ProcessEvent		event(RNotificationServicePolicy1Event1C0)
Policy2	ProcessEvent		event(RNotificationServicePolicy2Event1C0)
Policy3	ProcessEvent		event(RNotificationServicePolicy3Event1C1)
Policy4	ProcessEvent		event(RNotificationServicePolicy4Event1C0)
Policy1CU	RNotificationServicePolicy1Event1CU0		event(RNotificationServicePolicy1Event1aA) && event(RNotificationServicePolicy1Event1bA)
Policy2CU	RNotificationServicePolicy2Event1CU0		event(RNotificationServicePolicy2Event1aA) && event(RNotificationServicePolicy2Event1bA)
Policy3CU	RNotificationServicePolicy3Event1CU1		event(RNotificationServicePolicy3Event1aA) && event(RNotificationServicePolicy3Event1bA)
Policy4CU	RNotificationServicePolicy4Event1CU0		event(RNotificationServicePolicy4Event1aA)

Table 11, Enforceable policies generated for Notification Service

LoginService (AtomicProcess)			
Name	**Event**	**Condition**	**Action**
Policy1	RNotificationServicePolicy1Event1C0	member==Gold	event(RNotificationService Policy1Event1CU0)
Policy2	RNotificationServicePolicy2Event1C0	member==Bronze	event(RNotificationService Policy2Event1CU0)
Policy4	RNotificationServicePolicy4Event1C0	member==LeonClub	event(RNotificationService Policy4Event1CU0)

Table 12, Enforceable policies generated for Login Service

MessageService (AtomicProcess)			
Name	**Event**	**Condition**	**Action**
Policy3	RNotificationService Policy3Event1C1	priority==high	event(RNotificationService Policy3Event1CU1)
Policy2C	RNotificationService Policy2Event1aA		event(ShortMessageModeEvent)
ShortMessageModePolicy1	ProcessEvent&& ShortMessageModeEvent		setShortMode()
Policy4C	RNotificationService Policy4Event1aA		event(MessageInFrenchEvent)
MessageInFrenchPolicy1	ProcessEvent&& MessageInFrenchEvent		setLanguage(French)

Table 13, Enforceable policies generated for Message Service

AddressBookService (AtomicProcess)			
Name	**Event**	**Condition**	**Action**
Policy1C	RNotificationServicePolicy1Event1aA		event(QuickSearchEvent)
QuickSearchPolicy1	ProcessEvent&& QuickSearchEvent		doQuickSort()&& event(QuickSearchState1Event)
QuickSearchPolicy2	SortedEvent&& QuickSearchState1Event		doBinarySearch()
Policy2C	RNotificationServicePolicy2Event1bA		event(SimpleSearchEvent)

SimpleSearchPolicy1	ProcessEvent&& SimpleSearchEvent		doSimpleSort()&& event(SimpleSearchState1Event)
SimpleSearchPolicy2	SortedEvent&& S impleSearchState1Event		doBinarySearch()
Policy3C	RNotificationServicePolicy3Event1aA		event(EmergencyContactEvent)
EmergencyContact Policy1	ProcessEvent&& EmergencyContactEvent		useEmergencyContact()

Table 14, Enforceable policies generated Address Book Service

ContactService (CompositeProcess)			
Name	**Event**	**Condition**	**Action**
Policy1C	RNotificationServicePolicy1Event1bA		event(ContactServicePolicy1CEvent2aA)
Policy3C	RNotificationServicePolicy3Event1bA		event(ContactServicePolicy3CEvent2aA)

Table 15, Enforceable policies generated for Contact Service

PhoneService (AtomicProcess)			
Name	**Event**	**Condition**	**Action**
Policy3CC	ContactServicePolicy3CEvent2aA		event(AuthenticationEvent)
AuthenticationPolicy1	ProcessEvent&& AuthenticationEvent		setAuthentication()

Table 16, Enforceable policies generated for Phone Service

EmailService (AtomicProcess)			
Name	**Event**	**Condition**	**Action**
Policy1CC	ContactServicePolicy1CEvent2aA		event(RichContentEvent)
RichContentPolicy1	ProcessEvent&& RichContentEvent		doHtmlMessage()&& event(RichContentState1Event)
RichContentPolicy2	MessageCreatedEvent&& RichContentState1Event		doHmtlHeader()

Table 17, Enforceable policies generated for Email Service

Again, in order to validate these policies, the adaptive web services described above were implemented as shell web services, and they were hosted by the policy evaluation framework together with these enforceable policies. The produced enforceable policies were transformed into jess rules and executed by the framework's rule engine to manage these adaptive web services. Figure 5-8 shows a snippet of these policies as jess rules; in particular it illustrates the jess rules for the third management policy – policy for high priority notifications.

Some of the adaptive behaviours are more complex than others in this case study; the complexity is in the sense of the number of states it has in its FSM model, i.e. the number of actions that needs to be performed. The adaptive behaviours QuickSearch and RichContent are such adaptive behaviours. Therefore, it can be noted that the enforceable policies generated for the Address Book and Email services, where these adaptive behaviours reside, are invoking all the necessary actions and account for all the events and conditions defined in their adaptive behaviour's FSM model.

```
(defrule MessageServicePolicy3
(event (service ?service0&MessageService|Any) (type ?type0) (name
?event0&RNotificationServicePolicy3Event1C)) (event (service
?cservice0) (type ?ctype0) (name ?cevent0&~IdleEvent))
(param (service ?cservice0&MessageService) (direction
?direction0) (name ?param0&priority) (value ?value0)) (test (eq
?value0 high))
 =>(assert (event (service Any) (type Policy) (name
RNotificationServicePolicy3Event1CU1))) (assert (action (service
?service0) (name ?event0))))

(defrule PhoneServicePolicy3CC
(event (service ?service0&PhoneService|Any) (type ?type0) (name
?event0&ContactServicePolicy3CEvent2A))
=>(assert (event (service Any) (type Policy) (name
AuthenticationEvent))) (assert (action (service ?service0) (name
?event0))))

(defrule RNotificationServicePolicy3CU
(event (service ?service0&NotificationService|Any) (type ?type0)
(name ?event0&RNotificationServicePolicy3Event1CU1))
=>(assert (event (service Any) (type Policy) (name
RNotificationServicePolicy3Event1A))) (assert (action (service
?service0) (name ?event0))))

(defrule AddressBookServicePolicy3C
(event (service ?service0&AddressBookService|Any) (type ?type0)
(name ?event0&RNotificationServicePolicy3Event1A))
=>(assert (event (service Any) (type Policy) (name
EmergencyContactEvent))) (assert (action (service ?service0)
(name ?event0))))

(defrule ContactServicePolicy3C
(event (service ?service0&ContactService|Any) (type ?type0) (name
?event0&RNotificationServicePolicy3Event1A))
=>(assert (event (service Any) (type Policy) (name
ContactServicePolicy3CEvent2A))) (assert (action (service
?service0) (name ?event0))))

(defrule RNotificationServicePolicy3
(event (service ?service0&NotificationService|Any) (type ?type0)
(name ?event0&ProcessEvent))
=>(assert (event (service Any) (type Policy) (name
RNotificationServicePolicy3Event1C))))
```

Figure 5-8, Jess rules for third management for Notification Service

The request made by Susan Smith as a gold subscriber to the Notification Service and the response returned can be seen in Table 18.

Request	Response

Parameter	Value
Username:	*Susan.Smith*
Password:	*1234*
Member:	*Gold*
Subject:	*Times*
Message:	*This month's edition of the Times magazine has arrived*
Priority:	*Low*
Recipient:	*John.Murphy*
Group:	*None*

Parameter	Value
Call response	*true*
Email response	*true*
Process time	*54*
Search time	*1046*
Language	*en_US*

Table 18, Gold user type request and response to the Notification Service

Another request made by Susan Smith for a high priority message to the same instance of the Notification Service and the response returned, shown in Table 19.

Request

Parameter	Value
Username:	*Susan.Smith*
Password:	*1234*
Member:	*Silver*
Subject:	*Parcel Delivery*
Message:	*Your package has arrived*
Priority:	*High*
Recipient:	*John.Murphy*
Group:	*None*

Response

Parameter	Value
Call response	*true*
Email response	*true*
Process time	*24*
Search time	*2000*
Language	*en_US*

Table 19, High priority type request and response to the Notification Service

It can be noted that the process time and the search time for the gold subscriber is faster than the silver subscriber. Since this logic is expressed through policies, a different service provider could use different schema for the subscribers, without having to re-implement these services.

Once the policy rules were loaded into the policy evaluation framework and the web services executed, runtime traces were recorded. Figure 5-9 shows the content of a runtime log file for the first request, where the first management policy comes to effect and it changes the service instance to perform a quick search and use rich content (html) for gold members. While Figure 5-10 shows the trace log of this service at runtime for the second request, where the third policy comes into effect and it changes the service instance to select emergency contact details, and request pin to authenticate the recipient when notifying them. Please note the difference in the trace logs where changes in the service behaviour are indicated with $$.

```
[Login] Reading user list from file
[Login] username = Susan.Smith
[Login] password = 1234
[Login] membership = Gold
[Login] Susan Smith has successfully login
[AddressBook] Reading user list from file
[AddressBook] Reading contact list from file
Receiving Actions
Action: doQuickSort
[AddressBook] Requesting search for contact for user John.Murphy
[AddressBook] $$Using Quicksort algorithm to sort contact list
Sending Event SortedEvent
Receiving Actions
Action: doBinarySearch
[AddressBook] $$Searching for 02 using Binary search algorithm
[AddressBook] Contact detail found for originator:
Susan.Smith@moto.co.uk, 9894455
[AddressBook] $$Searching for 01 using Binary search algorithm
[AddressBook] Contact detail found for recipient:
John.Murphy@svaley.com, 8381122
Sending Event MessageCreatedEvent
Receiving Actions
Action: doHtmlMessage
[Email] Creating email body
[Email] $$Adding HTML tags to email body
[Email] $$Transforming message to HTML format
[Email] Adding the following message to email body This month's
edition of the Times magazine has arrived
[Email] Generating email header
[Email] Setting From field = Susan.Smith@moto.co.uk
[Email] Setting To field = John.Murphy@svaley.com
[Email] Setting email size
Receiving Actions
Action: doHmtlHeader
[Email] Sending email to Susan.Smith@moto.co.uk
[Email] Email Acknowledgement received
[Phone] Creating vxml dialog to notify recipient by phone
[Phone] Calling recipient on 8381122
[Phone] User has answered the call
[Phone] Informing recipient of a notification from 9894455
[Phone] Notifying recipient with the following message: This
month's edition of the Times magazine has arrived
[Phone] User has acknowledged this message
```

Figure 5-9, Runtime trace for Notification service with gold member policy triggered

```
[Login] Reading user list from file
[Login] username = Susan.Smith
[Login] password = 1234
[Login] membership = Silver
[Login] Susan Smith has successfully login
[AddressBook] Reading user list from file
[AddressBook] Reading contact list from file
Receiving Actions
Action: useEmergencyContact
[AddressBook] Requesting search for contact for user John.Murphy
[AddressBook] $$Changing criteria to search for emergency contact
address
Sending Event SortedEvent
[AddressBook] Searching for 02 using Linear search algorithm
[AddressBook] Contact detail found for originator:
Susan.Smith@moto.co.uk, 9894455
[AddressBook] Searching for 01 using Linear search algorithm
[AddressBook] Contact detail found for recipient:
John.Murphy@svaley.com, 8381122
[Email] Creating email body
[Email] Adding the following message to email body Your package
has arrived
[Email] Generating email header
[Email] Setting From field = Susan.Smith@moto.co.uk
[Email] Setting To field = John.Murphy@svaley.com
[Email] Setting email size
Sending Event MessageCreatedEvent
[Email] Sending email to Susan.Smith@moto.co.uk
[Email] Email Acknowledgement received
Receiving Actions
Action: setAuthentication
[Phone] Creating vxml dialog to notify recipient by phone
[Phone] $$Changing the vxml dialog to request user's pin before
notifying recipient
[Phone] Calling recipient on 8381122
[Phone] User has answered the call
[Phone] $$Requesting the user for a pin
[Phone] $$Validating the pin supplied by the user - accepted
[Phone] Informing recipient of a notification from 9894455
[Phone] Notifying recipient with the following message: Your
package has arrived
[Phone] User has acknowledged this message
```

Figure 5-10, Runtime trace for Notification service with high priority policy triggered

It can be noted that the same web service was dynamically modified for these scenarios according to their specified high-level policies (**$$** indicates the changes in the service behaviour). For full detail of the artefacts used in this case study – see Appendix G.

This case study showed how management policies specified for an adaptive composite service can empower it to dynamically adapt to context changes. It also showed how the tools allow a high-level policy to be automatically refined in a hierarchical manner into low-level policies, which are generated for the relevant constituent web services to modify their behaviours, so as to accommodate change in business logic. Note that the composition of the Notification service differs from the web service used in the previous case study.

It showed how the SMPE tool allowed users to specify several management policies for the Notification service and was capable of automatically refining them into low-level policies for the relevant constituent adaptive atomic web services. It also showed how these enforceable policies managed to change the behaviour of the constituent services according to the specified policies. Furthermore, the enforceable policies did not conflict with each other and only the necessary policies were triggered for each request based on the conditions met. It also showed how the integrated tools generate the necessary enforceable policies according to their FSM description even for more complex adaptive behaviours such as QuickSearch and RichContent.

This case study demonstrated how web services can be modified to accommodate changes in business logic without having to recompile them. Instead, policies were used which allow web services to have their behaviours adapted to suit business logic during runtime. In addition it shows that by using this approach, the adaptive web services could be dynamically adaptable to context changes, as long as management policies could be specified for those circumstances.

5.3 SABE Tool Usability Evaluation Experiment

The aim of this usability evaluation experiment is to appraise the usability of the implemented prototype SABE tool. This usability evaluation measures the degree of ease with which users can complete specific tasks using this tool, and mainly focusing on user satisfaction. The usability evaluation experiment was designed with three different usability tests (see Appendix A), and since it was not possible to get a large sample of users in the same place at the same time, these usability tests were conducted over three usability evaluation cycles.

Each usability evaluation cycle was conducted with a group of users taken from Computer Science and Computer Engineering graduates. Thus, the evaluation assumed that the users had some knowledge of computer programming. Although these users are not web service or policy experts, they are in fact undergoing a course in those areas and therefore they were very much interested in performing this experiment. More importantly, these users are new to the area of creating web services, and therefore they do not have preconceptions and specific habits that might interfere with their evaluation of this tool.

The first usability evaluation cycle was conducted with a small population of users allowing for a one to one interaction, where users were observed individually during the experiments. Any minor issues encountered were resolved before the next cycles. The second and third usability evaluation cycles were conducted in a group manner with a larger population of users, so that the tool could be evaluated in a wider exposure. A chi square test was then performed to evaluate the similarity in opinion responses for user satisfaction between the user groups in order to establish an overall larger sample group (see Appendix B).

In order to design the usability tests for this experiment, experiment goals were first identified. Three experiment goals were identified (described below), that instructed the creation of a usability test for each of these goals. These goals were derived from the thesis' objectives, so as to validate the usability of the prototype SABE tool against the proposed approach. A preliminary set of questions were asked at the beginning of each evaluation cycle to determine the user‘s experience and knowledge in the areas of Web Services, Finite State Machines, and Adaptive Behaviours.

Furthermore, at the end of each usability tests was a questionnaire, to assess the user's usability experience when using the SABE tool. The questionnaire was broken down into questions tailored to establish the user's experience of the different aspects of this tool after performing typical tasks with it. The questionnaire covered areas such as ease of use, comprehension, and helpfulness. In this manner, it was possible to appraise the usability of each aspect of this tool, and conclude its level of usability.

Likert scale type questions [78] were used in the questionnaire with four possible answers, ranging from 'very difficult', 'difficult', 'easy', to 'very easy'. Thus, it encouraged the users to choose an answer carefully, and not consolidate with a neutral answer (middle option). Open questions were also provided, allowing users to express their opinions that could not be captured with close.

5.3.1 Goals of SABE Tool Usability Evaluation Experiment

To evaluate the usability of the SABE tool, the following experiment goals were identified:

1. To determine the usability of the SABE tool in displaying the descriptions of the adaptive behaviours of atomic services as Finite State Machines.
2. To determine the usability of the SABE tool in describing the adaptive behaviours of atomic services as Finite State Machines.
3. To determine the usability of the SABE tool in aggregating Finite State Machines describing the adaptive behaviours of constituent atomic service for adaptive composite services.

A usability test was designed for each of the three experiment goals identified above. Each usability test was designed with a description of tasks, instructions, sample services, and a questionnaire to validate the experiment goal in accordance to the user's experience. Additionally, a set of pre-test questions was designed to determine the user's knowledge in the areas of web service, finite state machine, and adaptive behaviours.

The questionnaire was tailored to query the user's usability experiences in the various aspects of this tool, in accordance with the experiment goal. This questionnaire was designed in a formal manner, by decomposing the experiment goals into the three areas of ease of use, comprehension, and helpfulness, as depicted in Figure 5-11. Questions for this questionnaire were tailored to satisfy each of these sub-goals. It also facilitated an analysis of the answers within those stated areas.

- ❖ To determine the usability of the SABE tool in displaying the descriptions of the adaptive behaviours of atomic services as Finite State Machines
  - ➢ Find how easy is it to view descriptions of adaptive behaviours
  - ➢ Find how understandable is the process of viewing descriptions of adaptive behaviours
  - ➢ Find how helpful is the tool in viewing descriptions of adaptive behaviours
- ❖ To determine the usability of the SABE tool in describing the adaptive behaviours of atomic services as Finite State Machines
  - ➢ Find how easy is it to describe adaptive behaviours
  - ➢ Find how understandable is the process of describing adaptive behaviours
  - ➢ Find how helpful is the tool in describing adaptive behaviours
- ❖ To determine the usability of the SABE tool in aggregating FSM describing the adaptive behaviours of constituent atomic service for adaptive composite services
  - ➢ Find how easy is it to aggregate adaptive behaviours of a composite web service
  - ➢ Find how understandable is the process of aggregating adaptive behaviours of a composite web service
  - ➢ Find how helpful is the tool in aggregating adaptive behaviours of a composite web service

Figure 5-11, Decomposition of the experiment goals for the SABE tool's usability evaluation

For each of the usability tests conducted, an analysis was performed of the answers provided by the users. The answers for the multiple-choice questions were grouped according to the areas of ease of use, comprehension, and helpfulness. A histogram was generated for these answers where the possible answers are: *very difficult, difficult, easy, or very easy*. The results were reduced to the nominal level by combining all *agree* and *disagree* responses into two categories of *easy* and *difficult*. Then pie charts were generated, which provided a graphical view of the answers and their percentages. A description of the tests performed and the result-analyses are presented for each usability evaluation in the succeeding sections.

5.3.2 Method of the SABE Tool Usability Evaluation Cycles

The usability tests were presented online (over the internet) to the users. In addition to the pre-test questions, each usability test presented background information on the concepts used in the experiment for users not familiar with them. In order to better acquaint the users, it also contained a description of the usability test and of the tool used. In addition, a link to the SABE tool was provided, as well as a sample of web service descriptions for the usability test, which could be downloaded from the web site and executed on their computer. A textual description of the web service sample to be used in the experiment and a set of tasks to be performed on them using the tool were also provided. Instructions on how to get started with each task were given also. Having completed these tasks, the users were asked to complete a questionnaire.

The first usability evaluation cycle was conducted with six users in an individual manner, where users were observed individually during the experiments. This evaluation cycle was performed in this manner to acquire better individual feedback from the users about the usability of the SABE tool. Explanations were provided for any queries the users might have about the experiment and the SABE tool.

The next two usability evaluation cycles were conducted with twelve different users each, in a group manner, over the period of an hour. This evaluation cycle was performed in this manner to get a broader usability exposure of the SABE tool. Feedback from the first usability evaluation cycle was taken into account, and some changes were made to these usability evaluation cycles.

Some minor changes were made to the SABE tool since a flaw in the table widget was observed during the first evaluation cycle. The table widget allows users to edit properties of a selected node. However, in order to execute these changes, the user has to either press the "enter" button or click somewhere on the table, otherwise the changes are not executed. This issue was observed early on in the first evaluation cycle. In order to address this issue, two small changes were made to this tool since the table widget is a Swing component (third party). Incomplete required properties are now highlighted in red, and a report is presented to the user of the model validation, which is performed during the saving process. Thus allowing the user to know if the model is incomplete and also helping the user to spot any non-configured required properties.

Some improvements were made to the usability tests for the second and third test cycle, after some minor shortcomings in the first evaluation cycle were spotted. The changes are as follows:

- A warning about pressing the "enter" button after editing a property value;
- Some re-wording and spell checking of the usability test description;
- Updates to some of the questions in the questionnaire which users found confusing;

The following preliminary set of questions were asked at the beginning of the usability evaluation to determine the user's experience and knowledge in the areas of Web Services, Finite State Machines, and Adaptive Behaviours:

Web Services
101. How would you rate your knowledge of Semantic Web Service (Owl-S)?
102. Have you studied about web services?
103. Have you used a web service?
104. Have you created a web service?
105. Have you modelled a web service semantically in Owl-S?
Finite State Machines
111. How would you rate your knowledge of Finite State Machine?
112. Have you studied about Finite State Machine?
113. Have you seen a piece of software described as a Finite State Machine?
114. Have you described a piece of software semantically?
115. Have you described a piece of software using Finite State Machine?
Adaptive Behaviours
106. How would you rate your knowledge of (software) Adaptive Behaviours?
107. Have you studied about (software) Adaptive Behaviours?
108. Have you used a piece of software with some adaptive behaviour?
109. Have you configured an adaptive piece of software or application?
110. Have you created a piece of software with some adaptive behaviour?

Table 20, Preliminary set of questions for the usability evaluation of SABE tool

The answers to these questions were analysed for all the users and the following tables were produced:

	Web Services		
Usability Cycle	**Novice**	**Knowledgeable**	**Expert**
First Cycle	19%	50%	31%
Second Cycle	44%	56%	0%
Third Cycle	33%	56%	11%

Table 21, Results of the preliminary set of questions for knowledge on Web Services

	Finite State Machines		
Usability Cycle	**Novice**	**Knowledgeable**	**Expert**
First Cycle	7%	77%	17%
Second Cycle	33%	45%	22%
Third Cycle	42%	57%	2%

Table 22, Results of the preliminary set of questions for knowledge on Web Finite State Machines

	Adaptive Behaviours		
Usability Cycle	**Novice**	**Knowledgeable**	**Expert**
First Cycle	7%	77%	17%
Second Cycle	33%	45%	22%
Third Cycle	42%	57%	2%

Table 23, Results of the preliminary set of questions for knowledge on Adaptive Behaviours

It was observed that most of the users were either knowledgeable or experts in the field of Web Services, Finite State Machines, and Adaptive Behaviours. This is due to the fact that these users are undergoing a master course in computer science. Furthermore, it can be noted that the users from the first cycle are overall more knowledgeable in the enquired areas since these users were selected from postgraduates undergoing a PhD in Computer Science knowledgeable in such areas.

Three usability tests were conducted for each user to evaluate the usability of the SABE tool. Each usability test asked the users to perform a specific set of tasks so that the usability of the various aspects of this tool could be evaluated. Below is a summary of the tasks the users had to undertake in their evaluation tests.

SABE Usability Test 1 – "To determine the usability of the SABE tool in displaying the descriptions of adaptive behaviours of atomic services as Finite State Machines".

In this usability test, the users were asked to use the SABE tool to view the details of a set of adaptive behaviour descriptions, described as FSM, pertaining to an adaptive Print service. Users were asked to use the SABE tool to observe the parameters of this service, and more importantly the details of the adaptive behaviours. They were asked to observe the details of the FSM describing these adaptive behaviours: states and their actions, transitions and their triggers as well as their guards.

SABE Usability Test 2 – "To determine the usability of the SABE tool in describing adaptive behaviours of atomic services as Finite State Machines".

Similar to the previous evaluation cycle, in this usability test, the users were asked to use the SABE tool to define adaptive behaviour descriptions as FSM for an atomic web service, called PhotoService. This service is an adaptive photo processing service. The adaptive behaviours were: BlackWhitePhoto for processing the photo as black & white; RemoveRedeye for removing redeye from photos; HighPhotoQuality for processing the photos in high resolution.

SABE Usability Test 3 – "To determine the usability of the SABE tool in aggregating Finite State Machines describing the adaptive behaviours of constituent atomic services for a composite service".

In this usability test, the users were asked to use the SABE tool to describe adaptive behaviours, as FSM, of a composite service called PhotoAlbumPrintService. This service is composed of a PhotoService, a PhotoAlbumService, and a PrintService. Since the adaptive behaviours of a composite service is the resultant aggregation of the adaptive behaviours pertaining to its constituent services, this task is accomplished by using the SABE tool to automatically aggregate the adaptive behaviours previous described.

Analyses of the results for each of these usability tests in relation to effectiveness, efficiency, satisfaction are presented in the succeeding sub-sections.

5.3.3 Effectiveness Result Analysis of the SABE Tool

The usability evaluation requested that users utilised the SABE tool to perform specific tasks for each usability test. To measure the effectiveness of the usability evaluation, the accuracy and completeness were observed for each of the users performing these tasks. In other words, the effectiveness of a usability evaluation can be measured by the number of correct answers provided by the users after performing each task.

The first usability evaluation cycle was performed in an individual manner, where user's progress was monitored and any queries were dealt with immediately. It was observed that the users completed all their tasks, and that each of their tasks were performed correctly. While assistance was provided whenever requested, the number of assistance requests was not counted. However, the request for assistance while performing a task provided feedback to the usability evaluation and improvements were made in the instructions for the second and third usability evaluation cycles. The improvements were done to guarantee a better rate of success when performing the usability evaluation in a group manner.

For the second and third usability evaluation cycles, the users were asked the following questions at the end of the first and third usability tests, so as to attain the user's effectiveness about their usability tasks:

Questions with respect to the usability tasks
121. What is the activity of the ExpensiveModestate1 of the ExpensiveMode adaptive behaviour?
122. What is the type of the of the ExpensiveMode adaptive behaviour?
123. What is the guard condition in the transition (between ColourPrintingState1 and ColourPrintingState2) of the ColourPriting adaptive behaviour?
124. What triggers the transition (between ColourPrintingState1 and ColourPrintingState2) of the ColourPriting adaptive behaviour?
321. Does the FSM belonging to the composite service have IdleState, InputState, ProceState, OutputState like of an atomic service?
322. Are all the adaptive behaviours from the constituent services present in the FSM of the composite service?
323. How many adaptive behaviours are present under the ProcessState of the composite service's FSM?

Table 24, Task questionnaire for the usability evaluation of SABE tool

The effectiveness of the second usability test was evaluated by analysing the artefacts produced by the users. The artefacts were examined for specific values that should have been set based on the usability tasks.

Analysing the answered questions and the artefacts produced, it was noted that there was 100% completion for all the usability tests. In relation to accuracy, the answers to these questions and the artefacts were verified for all the users and the table below was produced in which the following conclusions were drawn:

Accuracy	Second Usability Cycle		Third Usability Cycle	
Usability Tests	*Right*	*Wrong*	*Right*	*Wrong*
First Test	96%	4%	96%	4%
Second Test	94%	6%	89%	11%
Third Test	100%	0%	89%	11%

Table 25, Degree of accuracy for SABE tool from second and third usability cycles

The usability evaluation resulted in a completion rate of 100% and an accuracy rate of no less than 89% for perform the tasks with the SABE tool.

Although, the accuracy rate of the second usability cycle yield a better result. These results provide a positive indication that the majority of the users were well capable of understanding and performing the tasks set out to them. It can also be concluded that the majority of the users did manage to use the SABE tool to correctly perform the tasks.

5.3.4 Efficiency Result Analysis of the SABE Tool

The usability evaluation was presented online with instructions at the beginning of the test and questionnaire that had to be answered online at the end of the usability test. Having the usability test online, it allowed for the users to be timed over each usability test. The timer would start when the usability test web page was opened and stop when the questionnaire was answered. However, this timer malfunctioned and most users' test did not get their time recorded.

Due to the lack of data, no emphasis can be placed on the efficiency of the SABE tool. Although the times were not recorded for each usability test, the entire usability evaluation for the SABE tool was performed in just under an hour by the users. Having three separate usability tests with multiple tasks completed and questionnaire answered in just under an hour, it can be concluded that the usability evaluation was completed in a reasonable time, and that the SABE tool has performed in an efficient manner for the users to have completed the tasks in that time.

5.3.5 Satisfaction Result Analysis of the SABE Tool

The questionnaire provided at the end of each usability test had questions tailored to access the user's satisfaction towards the SABE tool. Presented below are the set of questions and a result analysis of their answers.

<u>SABE Usability Test 1 – "To determine the usability of the SABE tool in displaying the descriptions of adaptive behaviours of atomic services as Finite State Machines".</u>

At the end of the usability test, the users were asked the following questions in their questionnaire:

Questions with respect to Easiness
128. Browsing the service's Finite State Machine (FSM) was?
129. Identifying the adaptive behaviours contained in the FSM was?
130. Viewing the details of an adaptive behaviour was?

131. Identifying the actions performed by an adaptive behaviour was?
132. Identifying the sequence of actions performed by an adaptive behaviour was?
Questions with respect to Comprehension
136. To understand the description of adaptive behaviours using SABE tool was?
137. To understand the description of states describing an action of adaptive behaviours using SABE tool was?
138. To understand the description of transitions describing a sequence of actions of adaptive behaviours using SABE tool was?
Questions with respect to Helpfulness in guiding users
133. Sabe tool made the process of browsing the description of an adaptive behaviour?
134. Sabe tool made the process of identifying the actions of an adaptive behaviour?
135. Sabe tool made the process of identifying the sequence of actions of an adaptive behaviour?

Table 26, Questionnaire for the first usability test of SABE tool

The answers to these questions were analysed for all the usability cycles and the table below was produced in which the following conclusions were drawn:

	Easiness		**Comprehension**		**Helpfulness**	
Usability Cycle	*Easy*	*Difficult*	*Easy*	*Difficult*	*Easy*	*Difficult*
First Cycle	77%	23%	78%	22%	89%	11%
Second Cycle	92%	8%	67%	33%	97%	3%
Third Cycle	92%	8%	78%	22%	88%	12%

Table 27, Degree of satisfaction in viewing adaptive behaviours from three usability cycles

Positive results such as these would indicate that the GUI designed for this tool was successful in displaying the adaptive behaviours, as FSM, of a web service. It shows that the combination of a tree widget - used to display adaptive behaviours belonging to a service - and a table widget – used to display the details of each node of this tree widget - was successful in displaying the adaptive behaviours of a service. These results also give a good indication of how well FSM used for describing adaptive behaviours of web services was accepted by users, since its lexical form was hidden from them.

SABE Usability Test 2 – "To determine the usability of the SABE tool in describing adaptive behaviours of atomic services as Finite State Machines".

Similar to the previous evaluation cycle, at the end of the usability test, the users were asked the following questions in their questionnaire:

Questions with respect to Easiness
221. Generating a FSM for the service's process was?
222. Creating an adaptive behaviour was?
223. Creating the states for this adaptive behaviour was?

224. Creating the transitions for this adaptive behaviour was?
225. Describing an adaptive behaviour was?
226. Describing the activities of an adaptive behaviour using states was?
227. Describing the sequence of activities for an adaptive behaviour using transitions was?
Questions with respect to Comprehension
232. To understand the process of generating a FSM was?
233. To understand the process of describing an adaptive behaviour was?
234. To understand the process of describing the activities of an adaptive behaviour as states was?
235. To understand the process of describing the sequence of activities for an adaptive behaviour as transitions was?
Questions with respect to Helpfulness in guiding users
228. Sabe tool made the process of generating a FSM for the service?
229. Sabe tool made the process of describing an adaptive behaviour?
230. Sabe tool made the process of describing the activities of an adaptive behaviour?
231. Sabe tool made the process of describing the sequence of activities for an adaptive behaviour?

Table 28, Questionnaire for the second usability test of SABE tool

The answers to these questions were analysed for all the usability cycles, and the table below was produced in which the following conclusions were drawn:

	Easiness		**Comprehension**		**Helpfulness**	
Usability Cycle	*Easy*	*Difficult*	*Easy*	*Difficult*	*Easy*	*Difficult*
First Cycle	89%	11%	75%	25%	97%	3%
Second Cycle	88%	12%	79%	21%	94%	6%
Third Cycle	87%	13%	79%	21%	88%	12%

Table 29, Degree of satisfaction in describing adaptive behaviours from three usability cycles

These results are even higher than the first usability test, suggesting how easy it is to describe adaptive behaviours as FSM with the SABE tool. It is logical to presume that once the users got used to using the tool, they found it easier and more helpful in describing the adaptive behaviours, although it is considered to be slightly more complex than viewing adaptive behaviours. These results also give a good indication of how well-accepted FSM was as a means to describe the adaptive behaviours of atomic services. By hiding the FSM lexical model, users who are novice to FSM were found to be able to describe adaptive behaviours with ease.

SABE Usability Test 3 – "To determine the usability of the SABE tool in aggregating Finite State Machines describing the adaptive behaviours of constituent atomic services for a composite service".

At the end of this usability test, the users were asked the following questions in their questionnaire:

Questions with respect to Easiness
324. Aggregating the adaptive behaviours for a composite service was?
325. Viewing the resultant aggregated adaptive behaviours was?
Questions with respect to Comprehension
328. To understand the process of aggregating the adaptive behaviours for a composite service was?
329. To understand the resultant aggregated adaptive behaviours using the SABE tool was?
Questions with respect to Helpfulness in automation
326. Sabe tool made the process of aggregating the adaptive behaviours for a composite service?
327. Sabe tool made the process of browsing the resultant aggregated adaptive behaviours?

Table 30, Questionnaire for the third usability test of SABE tool

The answers to these questions were analysed for all the usability cycles, and the table below was produced in which the following conclusions were drawn:

	Easiness		**Comprehension**		**Helpfulness**	
Usability Cycle	*Easy*	*Difficult*	*Easy*	*Difficult*	*Easy*	*Difficult*
First Cycle	100%	0%	67%	33%	100%	0%
Second Cycle	79%	21%	71%	29%	96%	4%
Third Cycle	83%	17%	79%	21%	88%	12%

Table 31, Degree of satisfaction in aggregating adaptive behaviours from three usability cycles

These results indicate that while the users thought the process of aggregating the adaptive behaviours for adaptive composite services is somewhat difficult to understand, none of the users thought that it was difficult to accomplish this process using the SABE tool. The SABE tool allows users to aggregate these adaptive behaviours in an automated manner. Thus, these results indicate that automating this aggregation process was an appropriate decision during the design of this tool. Again, it is seen with these high scores how useful the automated aggregation process is for describing adaptive behaviours of composite services. It can be noted that while the process of aggregating the adaptive behaviours for composite services would have been difficult and somewhat tiresome, the SABE tool overcomes this issue with the automation of this process and proves itself to be useful with these high scores.

Comparison between the SABE Tool Usability Evaluation Cycles

Small changes were made to the usability tests after the first evaluation cycle. These changes were only improvements to the tests, and did not affect the overall intent of these tests. Furthermore, the changes to the tool were minor changes, which had minimal to none improvement in the usability evaluation. Thus, a comparison of the results of the user's answers to the questionnaire was performed.

The answers for the various tests were grouped into the categories of: ease of use, helpfulness, and comprehension. A comparison was performed between the results of the evaluation cycles by applying a chi square test on the various answers. The chi square test mathematically compares the results of the evaluation cycles. By using the chi square test to compare these answers, it was found that the answers of all three usability tests passed the null hypothesis test thus, proving that there was no difference between the answers from the evaluations cycles. Consequently, this result strongly indicates that these three evaluation cycles concurrently provide the same results with regards to the usability evaluation of the SABE tool.

5.3.6 Overall Conclusion for the SABE Tool Usability Evaluation

After analysing the results of the usability tests performed on the SABE tool, conclusions were drawn based on their results. The first conclusion is that the results from these usability tests are a good indication that the SABE tool is easy to use, due to its ability to hide complex details of a FSM, such as its lexical form, and to provide a graphical user interface that users found helpful when describing adaptive behaviours as FSM.

Secondly, these results indicate that while the users thought that the process of aggregating the adaptive behaviours for a composite service was somewhat difficult to understand, none of the users believed it difficult to accomplish this process using the SABE tool. These results prove that automating this aggregation process was an appropriate decision during the design of this tool. It can be noted that while the process of aggregating the adaptive behaviours for composite services would have been difficult and somewhat tiresome, the SABE tool overcomes this issue with the automation of this process and proves itself to be useful with these high scores.

5.4 SMPE Tool Usability Evaluation Experiment

The aim of this usability evaluation experiment is to appraise the usability of the implemented prototype SMPE tool. This usability evaluation measures the degree of ease with which users can complete specific tasks using this tool, and mainly focusing on user satisfaction. The usability evaluation experiment was designed with three different usability tests (see Appendix C), and since it was not possible to get a large sample of users in the same place at the same time, these usability tests were conducted over three usability evaluation cycles.

In the same manner as the previous usability evaluation experiment performed on the SABE tool, each usability evaluation cycle was conducted with a group of users with some knowledge of computer programming. Pre-test questions were asked at the beginning of the evaluation cycle to determine their experience and knowledge in the area of policy-based management.

Again, objectives were first identified, before designing the usability tests for this experiment. Three experiment goals were identified, which instructed the creation of a usability test for each goal. These experiment goals were derived from the research objectives so as to validate the usability evaluation of the prototype SMPE tool.

In the same fashion as before, the usability tests were designed with a questionnaire at end of each of them to assess the user's usability experience when using this tool. The questionnaire was broken down into questions tailored to establish the user's experience of the different aspects of a tool, after performing typical tasks with this tool. The questionnaire covered areas such as ease of use, comprehension, and helpfulness. And the questionnaire was designed with four-point Liker scale type closed questions.

Once more, three usability evaluation cycles were conducted with the same set of users as was in the SABE's usability experiment. The first usability evaluation cycle was conducted with a small population of users in an individual manner, where users were observed individually during the experiments. The second and third usability evaluation cycles were then conducted in a group manner, with a larger population of users, so as to gain a larger sample group. Then a chi square test was conducted to evaluate the similarity between the results for user satisfaction of the different test cycles, in order to appraise them as a single large sample group (see Appendix D).

5.4.1 Goals of SMPE Tool Usability Evaluation Experiment

To evaluate the usability of the SMPE tool, the following experiment goals were delineated:

1. To determine the usability of the SMPE tool in viewing the details of a policy specified to manage adaptive composite services.
2. To determine the usability of the SMPE tool in specifying policy to manage the adaptive composite services.
3. To determine the usability of the SMPE tool in refining policies managing composite services to auto-generated (discrete) policies managing their constituent adaptive services.

These experiment objectives have influenced the design of the usability tests from the creation of tasks and sample services, which allow users to experience the usability of the tool to the design of the questionnaire which validates the experiment objective in accordance to the user's experience. In order to design the questions for this questionnaire, the experiment objectives were first decomposed into the three areas of easiness, comprehension, and helpfulness, as depicted in Figure 5-12.

- ❖ To determine the usability of the SMPE tool in viewing the details of a policy specified to manage adaptive composite services
  - ➢ Find how easy is it to view descriptions of management policies
  - ➢ Find how understandable is the process of displaying descriptions of management policies
  - ➢ Find how helpful is the tool in displaying management policies
- ❖ To determine the usability of the SMPE tool in specifying policy to manage the adaptive composite services
  - ➢ Find how easy is to describe management policies
  - ➢ Find how understandable is the process of describing management policies
  - ➢ Find how helpful is the tool in describing management policies
- ❖ To determine the usability of the SMPE tool in refining policies managing a composite service to auto-generated (discrete) policies managing their constituent adaptive services
  - ➢ Find how easy is it to refine management policies assigned to a composite web service
  - ➢ Find how understandable is the process of refining management policies for a composite service
  - ➢ Find how helpful is the tool in refining management policies for a composite web service

Figure 5-12, Breakdown of the usability evaluation's objectives for the SMPE tool

A usability test was designed for each of the three experiment goals defined above. Each usability test was designed with a description of tasks, instructions, sample services, and a questionnaire to access the user's usability experience. Additionally, a set of pre-test questions was designed to query the user's knowledge in the area of policy-based management. The questionnaire was tailored to query the user's usability experiences in the various aspects of this tool, divided into the areas of ease of use, comprehension, and helpfulness.

For each of the usability tests conducted, an analysis was performed on the answers provided by the users. The answers for the multiple-choice questions were grouped according to the areas of ease of use, comprehension, and helpfulness. A histogram was generated for these answers, where the possible answers are: *very difficult, difficult, easy, or very easy*. The results were reduced to the nominal level by combining all *agree* and *disagree* responses into two categories of *easy* and *difficult*. Then pie charts were generated that provide a graphical view of the answers and their percentages.

5.4.2 Method of the SMPE Tool Usability Evaluation Cycles

The usability tests were presented online (over the internet) to the users. Besides the pre-test questions, each usability test presented background information on the concepts used in the experiment in case the users were not familiar with them. The background information also contained a description of the usability test and of the tool used. In addition, a link to the SMPE tool was provided, as well as a sample of web service descriptions for the usability test that could be downloaded from the web site and executed on a computer. Additionally, a description of the web service sample to be used in the experiment, and a set of tasks to be performed on them using the tool was provided. Instructions to get started with each task were also provided. Having completed these tasks, the users were asked to complete a questionnaire.

The first usability evaluation cycle was conducted with six users, in an individual manner, where users were observed individually during the tests. This evaluation cycle was performed in this manner to acquire better individual feedback from the users about the usability evaluation of the SMPE tool. Explanations were provided for any queries the users might have about the experiment and the SMPE tool.

The second and third usability evaluation cycles were conducted with twelve different users each, in a group manner, over a period of an hour. This evaluation cycle was performed in this manner to ensure broader usability exposure of the SMPE tool. Feedback from the first usability evaluation cycle was taken into account, and some changes were made to these usability evaluation cycles.

The same flaw encountered in the SABE tool, was also present in the SMPE tool. Thus, a similar action was taken, and minor changes were made to the SMPE tool, to highlight in red incomplete required properties, and report details of the validation of the policies to the user before saving them.

Once more, some improvements were made to the second and third usability tests, after some shortcomings in the first evaluation cycle were spotted. The changes are as follows:

- A warning about pressing the "enter" button after editing a property value;
- Some rewording and spell checking of the usability test description;

- Updates to some of the questions which users found confusing;

The following preliminary set of questions were asked at the beginning of the usability evaluation to determine the user's experience and knowledge with regards to Policy:

Policy
401. How would you rate your knowledge of Policy?
402. Have you studied or read about policies?
403. Have you seen a policy rule before?
404. Have you used policy before?
405. Have you created a policy before?

Table 32, Preliminary set of questions for the usability evaluation of SMPE tool

The answers to these questions were analysed for all the users and the table below was produced in which the following conclusions were drawn:

	Policy		
Usability Cycle	**Novice**	**Knowledgeable**	**Expert**
First Cycle	17%	40%	43%
Second Cycle	28%	64%	5%
Third Cycle	20%	80%	0%

Table 33, Results of the preliminary set of questions for knowledge on Policy

It was observed that most of the users were either knowledgeable or experts in the area of policy management. This is due to the fact that these users are undergoing a course in policy-based management. Furthermore, it can be noted that there are more experts in the field of policy management for the first usability cycle since these users were selected from postgraduates undergoing a PhD in Computer Science knowledgeable in such area.

Three usability tests were conducted with each user to evaluate the usability of the SMPE tool. Each usability test asked the users to perform a specific set of tasks to evaluate the usability of the various aspects of this tool. Below is a summary of the tasks the users had to undertake in their evaluation tests.

SMPE Usability Test 1 – "To determine the usability of the SMPE tool in viewing the details of a policy specified to manage adaptive behaviours of a composite service".

In this usability test, the users were asked to use the SMPE tool to view the details of a management policy authored to manage the adaptive behaviours of an adaptive print service. This management policy was created to modify this service's behaviour to print in EconomyMode when a document has over 50 pages. As before, users were asked to use the SMPE tool to browse the various aspects of this management policy, such as event, condition, and action aspects.

SMPE Usability Test 2 – "To determine the usability of the SMPE tool in specifying policy to manage the adaptive behaviours of a composite service".

In this usability test, the users were asked to use the SMPE tool to define a set of management policies to manage the adaptive behaviour of an adaptive composite web service called PhotoAlbumPrintService. This web service processes photos, creates an album document from these photos, and prints this album document.

Users were asked to describe a management policy that modifies this service's behaviour to apply the BlackWhitePhoto and HighPhotoQuality adaptive behaviours when service is processing landscape photos. Then users were asked to define a second management policy that modifies this service's behaviour to apply the RemoveRedEye adaptive behaviour when service is processing portrait photos.

Full instructions were provided for specifying the first management policy, while only minimum details were given for specifying the remaining management policies, so as to allow the users to think for themselves when defining them

SMPE Usability Test 3 – "To determine the usability of the SMPE tool in refining policies managing a composite service to auto-generated (discrete) policies managing its constituent services".

In this usability test, the users were asked to use the SMPE tool to automatically refine a management policy specified for an adaptive composite service, called PhotoAlbumPrintService. This web service processes photos, creates an album document from these photos, and prints this album document. A management policy was specified to modify this service's behaviour to apply the

RemoveRedEye and CreateCalendar adaptive behaviours when processing portrait photos.

The users were asked to use the SMPE tool to refine this management policy into auto-generated discrete policies that manage the adaptive behaviours of the relevant constituent atomic web services.

Analyses of the results for each of these usability tests in relation to effectiveness, efficiency, satisfaction are presented in the succeeding sub-sections.

5.4.3 Effectiveness Result Analysis of the SMPE Tool

The usability evaluation requested that users utilised the SMPE tool to perform specific tasks for each usability test. To measure the effectiveness of the usability evaluation, the accuracy and completeness were observed for each of the users performing these tasks. In other words, the effectiveness of a usability evaluation can be measured by the number of correct answers provided by the users after performing each task.

The first usability evaluation cycle was performed in an individual manner, where user's progress was monitored and any queries were dealt with immediately. It was observed that the users completed all their tasks, and that each of their tasks were performed correctly. While assistance was provided whenever requested, the number of assistance requests was not counted. However, the request for assistance while performing a task provided feedback to the usability evaluation and improvements were made in the instructions for the second and third usability evaluation cycles. The improvements were done to guarantee a better rate of success when performing the usability evaluation in a group manner.

For the second and third usability evaluation cycles, the users were asked the following questions at the end of the first usability test, so as to attain the user's effectiveness about their usability tasks:

Questions with respect to the usability tasks
421. What is the event triggering the management policy ServiceAdapt1Policy2?
422. What is the action the management policy ServiceAdapt1Policy2 will perform?
423. What is the condition the management policy ServiceAdapt1Policy2 must first satisfy?

Table 34, Task questionnaire for the usability evaluation of SMPE tool

The effectiveness of the second and third usability tests were evaluated by analysing the artefacts produced by the users. The artefacts were examined for specific values that should have been set based on the usability tasks.

Analysing the answered questions and the artefacts produced, it was noted that there was 100% completion for all the usability tests. In relation to accuracy, the answers to these questions and the artefacts were verified for all the users and the table below was produced in which the following conclusions were drawn:

Accuracy	**Second Usability Cycle**		**Third Usability Cycle**	
Usability Tests	*Right*	*Wrong*	*Right*	*Wrong*
First Test	97%	3%	89%	11%
Second Test	92%	8%	89%	11%
Third Test	100%	0%	100%	0%

Table 35, Degree of accuracy for SMPE tool from second and third usability cycles

The usability evaluation resulted in a completion rate of 100% and an accuracy rate of no less than 89% for perform the tasks with the SMPE tool. Although, the accuracy rate of the second usability cycle yield a better result. These results provide a positive indication that the majority of the users were well capable of understanding and performing the tasks set out to them. It can also be concluded that the majority of the users did manage to use the SABE tool to correctly perform the tasks.

5.4.4 Efficiency Result Analysis of the SMPE Tool

The usability evaluation was presented online with instructions at the beginning of the test and questionnaire that had to be answered online at the end of the usability test. Having the usability test online, it allowed for the users to be timed over each usability test. The timer would start when the usability test web page was opened and stop when the questionnaire was answered. However, this timer malfunctioned and most users' test did not get their time recorded.

Due to the lack of data, no emphasis can be placed on the efficiency of the SMPE tool. Although the times were not recorded for each usability test, the entire usability evaluation for the SMPE tool was performed in just under an hour by the users. Having three separate usability tests with multiple tasks and questionnaire completed in just under an hour, it can be concluded that the usability evaluation was done in a reasonable time, and that the SMPE tool has performed in an efficient manner for the users to have completed the tasks in such time.

5.4.5 Satisfaction Result Analysis of the SMPE Tool

The questionnaire provided at the end of each usability test had questions tailored to access the user's satisfaction towards the SMPE tool. Presented below are the set of questions and a result analysis of their answers.

SMPE Usability Test 1 – "To determine the usability of the SMPE tool in viewing the details of a policy specified to manage adaptive behaviours of a composite service".

At the end of the usability test, the users were asked in their questionnaire the following questions:

Questions with respect to Easiness
426. Viewing the details of the policy managing this service was?
427. Identifying what event triggers the policy was?
428. Identifying what condition the policy is evaluating was?
429. Identifying what adaptive behaviour is the policy is performing as its action was?
Questions with respect to Comprehension
434. To understand the description of the policy managing this service using SMPE tool was?
435. To understand the description of the event that triggers this policy using SMPE tool was?
436. To understand the description of the condition that must be satisfied for this policy using SMPE tool was?
437. To understand the description of the action that is performed for this policy using SMPE tool was?
Questions with respect to Helpfulness in guiding users
430. Smpe tool made the process of browsing the description of the policy managing this service?
431. Smpe tool made the process of browsing the description of the event that triggers this policy?
432. Smpe tool made the process of browsing the description of the condition that must be satisfied for this policy?
433. Smpe tool made the process of browsing the description of the action that is performed for this policy?

Table 36, Questionnaire for the first usability test of SMPE tool

The answers to these questions were analysed for all the usability cycles, and the table below was produced in which the following conclusions were drawn:

	Easiness		**Comprehension**		**Helpfulness**	
Usability Cycle	*Easy*	*Difficult*	*Easy*	*Difficult*	*Easy*	*Difficult*

First Cycle	96%	4%	83%	17%	100%	0%
Second Cycle	96%	4%	92%	8%	100%	0%
Third Cycle	94%	6%	88%	12%	96%	4%

Table 37, Degree of satisfaction in viewing policies from three usability cycles

These results give a good indication that the SMPE tool is capable of displaying the management policies of a web service successfully. In addition, they shown that the users were able to understand the management policies provided to manage a service through its adaptive behaviours. It was demonstrated that the combination of a tree widget and a table widget - used to display management policies and its details - work just as well as it did for the SABE tool in displaying adaptive behaviours.

SMPE Usability Test 2 – "To determine the usability of the SMPE tool in specifying policy to manage the adaptive behaviours of a composite service".

At the end of the usability test, the users were the following questions asked in their questionnaire:

Questions with respect to Easiness
521. Creating a policy for managing the service was?
522. Creating an event that would trigger this policy was?
523. Creating a condition that must be satisfied for this policy was?
524. Creating an action that would be performed by this policy was?
Questions with respect to Comprehension
529. To understand the process of creating a policy for managing the service using SMPE tool was?
530. To understand the process of describing the event that would trigger this policy using SMPE tool was?
531. To understand the process of describing the condition that must be satisfied for this policy using SMPE tool was?
532. To understand the process of describing the action that would be performed by this policy using SMPE tool was?
Questions with respect to Helpfulness in guiding users
525. Smpe tool made the process of creating the policy for managing the service?
526. Smpe tool made the process of describing the event that would trigger this policy?
527. Smpe tool made the process of describing the condition that must be satisfied for this policy?
528. Smpe tool made the process of describing the action that would be performed by this policy?

Table 38, Questionnaire for the second usability test of SMPE tool

The answers to these questions were analysed for all the usability cycles, and the table below was produced in which the following conclusions were drawn:

	Easiness	**Comprehension**	**Helpfulness**

Usability Cycle	*Easy*	*Difficult*	*Easy*	*Difficult*	*Easy*	*Difficult*
First Cycle	100%	0%	96%	4%	96%	4%
Second Cycle	90%	10%	94%	6%	92%	8%
Third Cycle	96%	4%	85%	15%	94%	6%

Table 39, Degree of satisfaction in authoring policies from three usability cycles

These results demonstrate how helpful and effective the SMPE tool is in authoring management policies for an adaptive composite service. It also demonstrates that by restricting the users with selections of appropriate vocabulary for different aspects of the policies, the tool facilitates the users in specifying policies, and it is considered helpful in preventing the user from authoring policies incorrectly. These high results also reveal how well suited the users found policies to be in managing the adaptive behaviours of a service, and how straightforward the process of authoring them with the help of the SMPE tool.

SMPE Usability Test 3 – "To determine the usability of the SMPE tool in refining policies managing a composite service to auto-generated (discrete) policies managing its constituent services".

At the end of this usability test, the users were asked the following questions in their questionnaire:

Questions with respect to Easiness
621. Refining policies for a service was?
622. Identifying the refined policies was?
623. Identifying the events of the refined policies was?
624. Identifying the conditions of the refined policies was?
625. Identifying the actions of the refined policies was?
Questions with respect to Comprehensiveness
625. To understand the process of generating refined policies to manage a service was?
626. To understand the resultant generated refined policies was?
625. To understand the process of generating refined policies to manage a service was?
626. To understand the resultant generated refined policies was?
625. To understand the process of generating refined policies to manage a service was?
Questions with respect to Helpfulness in automation
631. SMPE tool made the process of refining policies managing a service?
636. How helpful was SMPE in refining policies for a service?
632. SMPE tool made the process of identifying the refined policies?
633. SMPE tool made the process of identifying the events of the refined policies?
634. SMPE tool made the process of identifying the conditions of the refined policies?
635. SMPE tool made the process of identifying the actions of the refined policies?

Table 40, Questionnaire for the third usability test of SMPE tool

The answers to these questions were analysed for all the usability cycles, and the table below was produced in which the following conclusions were drawn:

	Easiness		Comprehension		Helpfulness	
Usability Cycle	*Easy*	*Difficult*	*Easy*	*Difficult*	*Easy*	*Difficult*
First Cycle	100%	0%	80%	20%	97%	3%
Second Cycle	96%	4%	88%	12%	96%	4%
Third Cycle	92%	8%	92%	8%	100%	0%

Table 41, Degree of satisfaction in refining policies from three usability cycles

When compared with the first usability test, it is observed that while the users have found the process itself easier after repeating the process of viewing the management policies, (even though it is to some extent different) they have found it more difficult to comprehend, since policy refinement is complex and the refined policies are very different from the original policy. Although users found the refinement process easy due to its automation, their understanding of the process was not in the same level. Thus, it can be seen that while the process of refining management policies for a composite service would have been very difficult and complex, the SMPE tool overcomes this problem with the automation of the refinement process. The SMPE tool's automated refinement process prevents users from making the mistakes they would make if they were to manually refine these policies. Thus it shows how valuable an asset the automated refinement process was to this tool. These results are a good indication that although refinement of policies is very complex, its automation, as it is performed by this tool, greatly facilitates its accomplishment.

Comparison between the SMPE Tool Usability Evaluation Cycles

Small changes were made to the usability tests after the first evaluation cycle. These changes consisted only of improvements to the tests, and did not affect the overall emphasis of these usability tests. Furthermore, the changes to the tool were minor changes which had minimal to no improvement in the usability. Thus a comparison was performed of the results of the user's answers to the questionnaire.

The answers for the various tests were grouped into the categories of: ease of use, helpfulness, and comprehension. A comparison was performed between the results of the evaluation cycles by applying a chi square test on the various answers.

By using the chi square test to compare these answers, it was found that the answers of all three usability tests passed the null hypothesis test, thus proving that there was no difference between the answers from the three evaluation cycles. Consequently, this result strongly indicates how well these three evaluation cycles are in agreement regarding the usability evaluation of the SMPE tool.

5.4.6 Overall Conclusion for the SMPE Tool Usability Evaluation

After analysing the results of the usability tests performed on the SMPE tool, conclusions were drawn based on their results. The first conclusion is that the results from these usability tests are a good indication that the SMPE tool is easy to use due to its graphical user interface, which users also found helpful when specifying management policies for these adaptive web services.

These results demonstrate how helpful and effective the SMPE tool is in authoring policies for adaptive web services. It also demonstrates that by restricting the users with appropriate selections for authoring the policies, the tool makes it very easy, and it is considered very helpful, in preventing the user from authoring syntactically incorrectly policies. These high results reveal how well-suited policies are to manage the adaptive behaviours of a service, and how easy it is to understand the process of authoring them with the help of the SMPE tool. However, it is important to note that the users in this usability evaluation were not professionals, and therefore their answers were not very critical.

Although users found the refinement process easy due to its automation, their understanding of the process was not at the same level. Thus, it can be seen that while the process of refining management policies for a composite service would have been very difficult and complex, the SMPE tool overcomes this problem with the automation of the policy refinement process. The SMPE tool's automated policy refinement process prevents users from making mistakes; such as if they were to manually refine these policies. Thus, it shows how useful and effective the automated policy refinement process was to this tool.

5.5 Related Work

Automated policy refinement has been the focus of several research projects and initiatives. This section illustrates some principal approaches, whilst differentiating them from our approach.

Verma [58] proposes a case-based reasoning approach to support policy refinement. In Verma's approach, the system learns experimentally from the operational behaviour it has previously examined. Each of those cases contains a combination of the system configuration parameters and the policy goals achieved. It takes the desired goal as input, and searches the case database to determine the optimal configuration that will satisfy this goal.

Due to its nature, Verma's approach could be easily adapted to refine policies to manage the adaptive behaviours of web services. It can observe expert users create high-level policies and the relevant configuration for managing adaptive web services.

The similarity between Verma's approach and our approach is that they both perform automated policy refinement. However they differ considerably in the way this is achieved and the circumstances under which it can be achieved. Verma's approach differs from our approach in the way that our tools seek to explicitly model only the adaptive behaviours and support empowerment of the administrators/managers in composing appropriate high-level management policies. These are then automatically refined, based on the modelling of web services and their adaptive behaviours.

Verma's approach is dependent on a rich enough case database, which is only made possible by observing the system for some period. Unlike our approach, such dependency limits this approach from automatically refining policies for new adaptive web services. Only if a policy expert first pre-configures the database with cases of every possible policy–configuration combination can this be achieved. Otherwise, a policy expert needs to be available to address the manual refinement of new policies.

Also, Verma's approach relies on observing users experience, which places a high burden on the expert manually mapping the high-level policies into enforceable policies. This can be a complex task when dealing with adaptive composite web services. Since the case database is populated by users, and it is not linked with the managed devices, it could contain incorrect configurations. This could perhaps occur due to misalignment with the managed system, which would then cause issues when employed. In our approach, policies are automatically refined, based on a rich semantic description of the managed adaptive web services. The automatic policy refinement process can be repeated when services are updated. These descriptions are normally described by the person responsible for creating the services; a person who is knowledgeable in the workings of their services.

Furthermore, existing cases in this policy refinement case database can be invalidated with the introduction of new configuration parameters, new service composition, or new conditions in the high-level policy. In our approach, the automated refinement process will automatically handle such changes. Since our policy refinement process is based on a rich semantic model, it can automatically generate updated policies based on the updated models.

Russo [53] presents a partially automated approach for policy refinement by which a formal representation of a system based on Event Calculus can be used in conjunction with abductive reasoning and goal elaboration techniques, to derive the sequence operations for a particular goal.

From the user's point of view, this approach is somewhat similar to ours. It expects a high-level policy (goal), and needs system behaviour description as state charts (like FSM) to automatically refined them into low-level policies. However, after a closer inspection, it can be noted that the two approaches are quite different.

The approach proposed by Russo requires system behaviour description as state charts, which can be complex and tedious to describe for large systems. In our approach, developers are only required to describe small aspects of the web services, which are deemed 'adaptive behaviours'. Thus, it encourages developers to adopt our approach by only describing some aspects of adaptive web services, as supposed to having to describe the entire system.

Russo's approach uses Event Calculus notation for its abductive reasoning. Hence, system behaviours are only described as state charts for convenience. This system description must be translated to Event Calculus notation before it can be used. In our approach, this transformation step is not needed, since policies are inferred directly from the FSM descriptions.

Although Russo's approach claims to be domain-independent, it is only useful if the policy refinement is performed in an automated manner. In order to be automated, it must first accumulate enough refinement patterns in a particular domain. These policy refinement patterns must be created manually, and this can be strenuous for the policy expert, especially if dealing with complex policies for large composite web services. In our approach, policies are always automatically refined. Furthermore, our approach restricts users from specifying unsupported high-level policies. This means that it can handle refining policies for new adaptive web services, without the need for a policy expert to be present. Thus, it promotes the employment of our approach by different web service providers, where their web service managers do not have to be policy experts or to have full knowledge of the intricacy of the managed web services.

Cassasa-Mont [70] outlines a policy-authoring environment that provides a policy toolkit, called POWER, for refining policies. A domain expert first develops a set of policy templates, expressed as Prolog programs, and the policy authoring tools have an integrated inference engine that interprets these programs to guide the user in selecting the appropriate elements from the management information model. These are then included in the final policy.

A major limitation of the approach proposed by Cassasa-Mont is that their system does not provide any support for automatically deriving the actions to be included in a policy template. A positive side to the Cassasa-Mont approach to policy refinement is that it can be adapted for different application domains. However, the templates must be manually defined by a domain expert in Prolog. Therefore, domain experts must have a detailed understanding of the system and formalism. Another limitation is that it also assumes that domain experts are knowledgeable in Prolog and so can create policy templates. In our approach, no templates are needed. Instead, automatic policy refinement is achieved by inferring directly from the adaptive web service's descriptions.

Kiel [3] suggests an automated policy refinement approach that accomplishes its refinement using Ontology-based service composition. By modelling each of the managed components/devices as Ontology-based web services using OWL-S, these components/devices can then be used by the policy refinement approach. The Kiel's approach is clever in dynamically discovering the policy refinement pattern using the web service composition (matchmaking engine). However, this approach will not always result in a policy refinement, since some of the required service might not be present for the service composition. The proposed approach will always result in a policy refinement since only valid high-level can be specified.

The Kiel approach uses service composition to generate our low-level mapping policies, and the low level enforceable policies are manually created and modelled as web services. In the proposed approach, mapping policies are generated based on a service composition, and enforceable policies are also automatically generated based on the adaptive behaviour description as FSM. In other words, the proposed approach takes one step forward and generates the low level enforceable policies as well.

Guerrero [2] presents a generic ontology-based policy refinement approach that also provides interoperability between high-level and low-level policies, by using Ontology models and SWRL rules. Guerrero's approach to policy refinement requires a very high modelling effort – modelling the system at every level as well as the relationship between them, together with translation rules. Too much effort is required for all this modelling, and unless the system is large and will not change for a long time, this effort is too great of a challenge for the outcome. In the proposed approach, policy refinement is inferred from a much smaller model without the need for translation rules.

Guerrero's approach assumes that policies will be manually specified at every level of abstraction. Thus it allows for interoperability between high-level and low-level policies. But this assumption adds an extra burden of having to manually create low-level policies. In the proposed approach, low level policies are seen as a realisation of the high-level policies and they should not be manually created. Such action also eliminates human error added in the creation of such policies.

Another approach, to automated policy refinement designed for security type policies comes from Albuquerque [65]. This is executed using a modelling technique where a system's model is structured in different abstraction levels. A system's objects, relationships, and policies at a certain abstraction level, together with the system model of the lower level and the relationship between entities of the two layers, enables the generation of lower level policies. It uses a Diagram of Abstract

Subsystem to model a management system (abstractly) segmented into Abstract Subsystems (AS). However, it doesn't support the refinement of obligation policies, since their approach only models RBAC systems and is not suitable for managing adaptive behaviours.

Another approach to automated policy refinement of security policies is from Cunningham [66]. It models the resource hierarchy, and this is used to refine policies assigned to the abstract resources in (on top of) the resource hierarchy, and automatically produce low-level policies for its concrete resources. Policies are first refined for a resource type, and then for its instance. It uses an AND/OR Graph to model resources, and an Arithmetic and Logical Expression Tree to write expressions modelling policy specification.

Rubio-Loyola [62] achieves policy refinement by breaking down high-level goals into AND/OR structures by using KAOS domain independent refinement patterns, as part of a goal graph elaboration step. An administrator then selects the most suitable lower level goals to be refined by the management module. Rubio-Loyola's approach is not automated, and it requires administrators to choose an appropriate set of refined policies. System behaviours of a managed system are modelled with labelled transition systems which are based on FSM. It suffers from scalability since it uses the FSM approach to model an entire managed system, without using a multi-layer of abstraction. In our approach, FSM is used only to describe the adaptive behaviours, and not the overall behaviour of (composite) web services. Because FSM is combined with OWL-S, it can be scaled for use with a large composite web service.

Table 42 presents a comparison of the approaches discussed. It focuses on comparing the key aspects of the policy refinement process.

	Carey	Russo	Verma	Power	Klie	Cunningham	Rubio-Loyola
Approach Kind	Modelling technique	KAOS goal elaboration & abductive reasoning	Case base reasoning & clustering techniques	Policy template	Web Service Composition	Inference of abstract hierarchy model	KAOS goal elaboration
Automated	Automated	Semi-automated	Semi-automated	Not automated	Automated	Automated	Not automated
Policy Type	Obligation	Goal	Configuration	Management	Obligation	Authorisation	Goal
Domain	Web Services	Network QoS	Network QoS	Network QoS	Network Management	Network Security	Network QoS
User Expertise Needed	System expert	System and Policy Experts	Policy or System Expert	Policy (Prolog) and System experts	System expert	System expert	System and Policy experts
Models used in Refinement	FSM & OWL-S	System behaviour as Event Calculus, domain hierarchy, KAOS patterns	Case database	Policy templates (Prolog)	OWL-S Model	DAOG & ALET	FSM, KAOS patterns
User Inputs	Adaptive Web Service description	State charts, domain hierarchy, KAOS patterns	Case entries	Policy templates	OWL-S Model	Resource type hierarchy	System behaviour
Outputs	Obligation Policy	Ponder policies	Configuration parameters	Configuration parameters	Configuration parameters	Policy as ALET	Ponder
Key Limitations	Applies only to Web service based systems	Need to cache patterns before being automated	Need to be pre populated with use cases	Need to create Prolog based templates	Need to have all the required web services available for composition	Need a detailed resource type hierarchy	Need to run simulation and manually select valid traces
Complexity	Describing the adaptive web services	Creating KAOS patterns	Creating use case entries	Creating the policy template	Describing low-level device policies as OWL-S web service	Describing the Resource type hierarchy	Analysing and choosing traces
Tool Support	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table 42, Policy refinement approaches comparison

5.6 Cost and Benefit of Adaptive Web Services

The proposed approach places a new view on web services; a vision where web services can adapt dynamically at runtime. This vision of adaptive web services provides many benefits to users and businesses alike, but for such flexibility there is a cost, and that cost is in the development of these services. In this section the cost and benefits of creating and managing adaptive web services are discussed.

There is an extra cost when developing adaptive web services compared to development of simple web services. With simple web services, the developers have to only concentrate in creating web services to do the task at hand. For adaptive web services, the developers have to also implement and describe the adaptive behaviours. Thus adding extra time and effort performing these tasks to develop adaptive web services. Furthermore, developers need to be aware of how web services will be used and what other needs the users of these web services might have in order to create these adaptive behaviours. However, this burden imposed on the developers pays off when they manage to foresee and implement the adaptive behaviours needed by diverse businesses or that it can satisfy several users' needs, i.e. adaptive behaviours which make their web services more flexible and therefore popular than other web services.

The MAWS methodology describes the process cycle of how adaptive web services are developed and managed, but more importantly it identifies the actors involved for each activity in the methodology process. The MAWS methodology identifies the web service developer as the actor responsible for implementing web services and describing their adaptive behaviours, while web service administrators are responsible for specifying policies to manage these adaptive web services. Furthermore, separate tools are provided to perform the different activities. By dividing the duties of development and management of adaptive web services to different actors it has the benefit of empowering the managers in modifying the web services to suit their needs, and thus freeing the developers from such burden.

5.7 Summary

This chapter demonstrated how well the novel prototype tools SABE and SMPE, behave under the different user cases scenarios presented. Furthermore it described in detail the usability tests performed on these innovative tools together with the results and evaluation of these experiments. Lastly this chapter presents a discussion on related work of automated policy refinement and adaptive composite services.

6 Conclusion

6.1 Introduction

In this chapter, the original objectives of the thesis are restated followed by a discussion of how these objectives were achieved. Then conclusions are drawn based on observations from the tool's design and results of the experiments performed. Limitations to the proposed approach are identified. Additionally, contributions made by the research presented in this thesis are described together with their benefits. Finally, some suggestions as to future enhancements of this research are explored.

6.2 Objectives and Achievements

The research goal driving this thesis was to propose and evaluate an innovative architectural approach and mechanism, which combines Finite State Machine (FSM) and Ontology reasoning together with policy-based management. This innovative approach supports accurate high-level policy specification and automatic refinement and generation of low-level policies for managing adaptive composite web services.

The objectives drawn from this research question were:

- Research the use of Finite State Machine (FSM) and ontological techniques to describe adaptive web services and techniques to support policy refinement.
- Define and develop innovative mechanisms to describe the adaptive behaviours of both atomic and composed web services.
- Define and develop novel mechanisms to specify high-level policies to manage the adaptive behaviours of composed web services and to auto-generate refined (low-level) policies, which can be enforced on the relevant constituent atomic services.
- Evaluate the complexity of designing adaptive service management using this approach, i.e. usability of the approach, and a comparison with other policy refinement approaches.

A discussion of how well these research objectives were achieved is presented below.

<u>“Research the use of Finite State Machine (FSM) and ontological techniques to describe adaptive web services and techniques to support policy refinement.”</u>

The key to success when managing adaptive web services is to have a semantically rich set of descriptions to reason about them and their adaptive behaviours. Ontology is being used more and more to semantically describe resources. Hence, OWL-S, an Ontology-based model for describing web services semantically, was found to be a semantically rich model chosen to describe web services and their composition. However, this Ontology model was not designed to describe the adaptive aspects of adaptive web services, i.e. their internal adaptive behaviours.

After investigating multiple modelling languages for representing behaviours of web services in a formal manner, FSM was identified as a strong modelling language candidate to formally describe the adaptive behaviours belonging to adaptive web services. FSM is a well-established model for formally describing a system’s behaviours, but there is a need to be careful of complexity when using this model. FSM uses states and transitions to describe behaviours, which when describing large and complex systems, can become convoluted, and difficult to understand.

However, if FSM is used judiciously, in what it describes and how it describes it, this complexity issue can be reduced or avoided. As shown in the proposed approach, by using FSM to describe only the adaptive behaviours of web services and not their entire behaviour, and by encapsulating each of them in sub-FSM, it was possible to significantly reduce the number of states and inter-relationships that needed to be described, thereby avoiding the complexity issue. Still, care needs to be taken in providing enough tools so that users can easily make use of FSM to describe adaptive behaviours pertaining to web services.

"Define and develop innovative mechanisms to describe the adaptive behaviours of both atomic and composed web services."

By describing adaptive behaviours pertaining to web services as FSM, users can reason about them in a formal manner, allowing policies to be specified to manage adaptive web services. However, using the FSM models manually in their raw form (lexical form) can be very tedious to write and complicated to read. Furthermore, it allows for syntactical errors to be introduced into the FSM description, therefore degrading its usefulness for reasoning. Thus, there is a necessity to build tools that actually hide the complexity of FSM and provides support in what needs to be described.

A set of integrated tools was developed and was fitted into a broader methodology, which gave users methodology steps, and showed how these novel tools can be used to support those steps. The WSDE and SABE tools were developed to shield the users away from the low-level details of OWL-S and FSM by hiding their lexical form, and to provide a method to visualise the OWL-S and FSM model using a graphical user interface. The SABE tool in particular facilitates the users in describing adaptive behaviours as FSM by having checks in place to aid the users in creating the FSM model and to prevent them from making structural mistakes in the FSM model.

Although it is possible for a developer to describe the adaptive behaviours of composite services by hand, this process is very tedious and error prone. Thus one of the requirements for developing the SABE tool was the ability to automatically aggregate the adaptive behaviours of the constituent services. Experiments performed on the SABE tool found it able to perform the aggregation of adaptive behaviours. But, the experiments were limited to sequential composition. It is believed that minor issues would arise for other types of composition.

Another issue is that the Ontology model OWL-S - used to describe web services - is a separate technology to the FSM model - used to describe their adaptive behaviours - and there is no natural bridge between the two. Therefore, there might be an issue with using the two of them together to describe adaptive web services.

This technological gap between OWL-S and FSM presented an initial challenge to the development of the SABE tool. However, this challenge was overcome by creating a FSM Ontology model (see Appendix E), which could be interpreted by the same Ontology reasoner as was used to reason about the web service's semantic description.

<u>"Define and develop novel mechanisms to specify high-level policies to manage the adaptive behaviours of composed web services and to auto-generate refined (low-level) policies, which can be enforced on the relevant constituent atomic services."</u>

A methodology, called MAWS methodology, was created to identify the set of steps required before management policies could be refined into low-level policies to manage adaptive web services. This methodology also helped identify a set of requirements for designing the SMPE tool; a tool for specifying management policies and for automatically refining them into auto-generated enforceable policies for managing adaptive web services.

The SMPE tool was designed with a graphical user interface that facilitates the users in authoring management policies, while hiding the complexity of such policies and of the adaptive web service's semantic description. This tool allows users to specify high-level policies, in which their vocabularies are based on abstract terminology, such as the name of a FSM representing an adaptive behaviour. However in order to manage these adaptive web services, these policies need to be refined.

Questions were raised as to whether the authoring of a high-level policy for managing an adaptive web service should be permitted when the web service does not have the resources to support the refinement of such policy. Thus, this tool prohibits users from expressing an out of scope management policy. Furthermore, it was found that the automated refinement and auto-generation of management policies into discrete enforceable policies could be assured by limiting their expressiveness to that provided by the semantic description of the adaptive web service.

By shielding the user from the intricacy of policy refinement and automating such a complicated and strenuous process, it was realized that non-policy experts can use the SMPE to specify high-level policies for managing adaptive web services. This tool can then generate low-level policies that enforce the goal of the specified policies. Unfortunately, users are currently not restricted from specifying policies that are semantically contradictive. Nor are they restricted from refining them. In other words, the enforceable low-level policies are correctly refined according to the intention of the specified high-level policy, even if semantically the specified policy does not make sense.

During the implementation of the SMPE tool, two sets of algorithms were developed for refining management policies in an automated manner. The first algorithm developed automatically refines only the action aspect of management policies; it is used for orchestration-type services, specifically for non-virtual composite web services. The second algorithm implemented automatically refines both the action and condition aspects of management policies; it is used for choreography-type composite web services. Both policy refinement algorithms make use of the event aspects of these policies in order to link high-level management policies to refined mapping policies, and to link mapping policies to low-level enforceable policies. Management policies could be linked to higher-level business policies through their event aspect.

"Evaluate the complexity of designing adaptive service management using this approach, i.e. usability of the approach, and a comparison with other policy refinement approaches."

The usability and effectiveness of the implemented tools were evaluated by undergoing a series of case studies to test their functionality, and a suit of usability experiments to test their ease of use. Case studies were designed to demonstrate the usefulness of the integrated tools in creating policies to manage different adaptive web services, and thereby validating the approach proposed in this thesis. A small web service rule engine had to be implemented together with shell adaptive web services in order to demonstrate the refined policies in effect.

Two case studies were conducted to demonstrate how effective these tools were in creating policies to manage:

i. a personalised holiday service, according to users preference;

ii. a notification service, according to context changes and business needs;

Several usability evaluations were performed on the innovative tools SABE and SMPE to validate their ease of use. The results from the usability experiments gave a strong indication of how easy and helpful these tools were in both describing adaptive web services, and specifying and refining management policies. Even though the overall results were good, there were some problems encountered with how the table widget was used by these tools. Although the context-sensitive dropdown menus used by these tools were found helpful in assisting users with the various tasks, it was realised that without an initial instruction provided for them, their operation can be obscure. It is granted that these usability evaluations were not conducted with professional developers or professional web service administrators.

Unfortunately, it was not possible to conduct the usability evaluations with professional web service or policy administrators, who most likely would have been more critical about them. However, the positive results provided by the usability experiments conducted with computer science graduates gave a good indication that any user with a background in computer programming and some understanding of web services and policies could use this tool with ease to specify policies to manage adaptive composite services.

Instead of performing a single usability evaluation with a large population, it was decided to perform an iteration of three evaluations, where the first evaluation is performed with a small group of users on a one to one basis. This provided fast feedback from the users and assessment of the evaluation progress, which were used to refine the usability tests for the next evaluations and fix any minor issues encountered so as to have successful subsequent evaluations. Then the second and third evaluations were performed in a group manner, with a larger group to grant a greater usability exposure. This approach to usability evaluation proved effective since some issues were encountered with the tool's GUI and small changes were made to improve them before the subsequent evaluations. If the usability evaluation was performed as a single evaluation, this minor issue could have badly affected the overall result of the evaluation and it would have been too late to resolve it.

A comparison analysis was performed over the results from all the evaluations, and it was found that, although the usability evaluations were performed with a small population, the pools of users had the same overall opinion about the usability of these tools.

The novel automated policy refinement process implemented in the SMPE tool was compared to other policy refinement processes performed by various research groups. It was concluded from this investigative comparison that none of other approaches have tried to use a model that combines FSM and OWL-S to reason about the policy refinement process in an automated manner.

It was confirmed that the SABE tool is very capable of successfully allowing users to describe adaptive behaviours pertaining to adaptive web services with ease. Furthermore, it was verified that the SMPE tool does perform well in specifying management policies and automatically refining them. The automation of the policy refinement and the provision of correct vocabulary for the policy specification were found to be the cornerstone to the approach proposed.

6.3 Limitations

However, the approach does have a number of limitations which need to be addressed in our future work. These can be summarised as follow:

- The proposed policy refinement approach does not provide parameter values for management policies: At present the enforcement policies produced from the policy refinement process only contain the operations that will achieve a particular goal, not the parameter values to be used with these operations.
- This approach excludes services with effects: The current solution is limited to managing a subset of adaptive web services, i.e. adaptive web services that do not account for changes in their effects. This approach was designed to only manage adaptive web services with fixed input and output types.
- Adaptive behaviours are not semantically filtered: At present the adaptive behaviours presented for a composite service is the resultant aggregation of the adaptive behaviours pertaining to its constituent services, without any constraints. There is neither semantic tagging nor filtering for the adaptive behaviours belonging to a web service.

Despite these limitations, the work presented in this thesis advances the current state of the art for managing adaptive web services and automatic policy refinement/generation. These contributions are described in the following section.

6.4 Contribution

This thesis proposes a novel model that combines FSM and Ontology reasoning, for describing adaptive composite services. This novel model, together with management policy, and an automated policy refinement technique are used to generate enforceable policies for managing these adaptive services. Although each of the technologies used by this model has been individually used in some form or another to describe a system's behaviours or to manage systems, this approach provides the first novel integration of these technologies to manage adaptive composite services.

This unique model provides support for an automated policy refinement technique. And this novel policy refinement approach was designed with the purpose of managing adaptive composite services by means of automatically refining specified high-level policies into generated discreet low-level policies, which directly control the behaviour of the relevant atomic services according to the intention of the specified high-level policy.

By using this novel approach, it is possible to remove complex business logic from the workflow[8] of composite web services and contain them within their constituent web services as adaptive behaviours that can be managed by policies. In this manner, composite web service's workflow can be kept simple, and hence fast, while businesses are still empowered with the flexibility to change the services behaviour using policies to suit their business goal. This is an example of web services adapting to context information.

As business requirements change, current web services need to evolve while trying to maintain a backward compatibility with previous client applications. Trying to support different versions of the same service can be very troublesome. Another way to address this issue is to model the new features for these services as adaptive behaviours and use policies to control when to enable the new feature.

Another benefit in adopting this innovative approach to web services is the individualisation empowerment that it provides to composite web services, while

[8] Complex logic written within workflow scripts or within the composition graph used to generate the composition workflow

keeping their workflow logic generic. Consequently it is kept efficient, and error-free. An example of this empowerment is demonstrated when businesses enable users to configure these web services based on their user's preferences.

A minor contribution to the state of the art in adaptive web service management is a set of integrated tools that embodies this model and approach. This set of tools hides the complexity of both modelling of the FSM and Ontology, and eases the complexity of authoring policies to manage the adaptive behaviours of composite services. In addition, it has the ability to automatically refine high-level policies and automatically generate judicious low-level policies.

Using the approach proposed in this thesis, together with the innovative prototype tools, allows administrators to create low-level policies to manage adaptive web services without the need to know the intricacy details of all pertaining adaptive behaviours. Furthermore, since these tools provide the users with a valid vocabulary when specifying management policies, and automatically refine them into auto-generated enforceable policies, it removes human error. Thus the creation of discrete low-level policies that conforms with the goals of the specified policy is assured.

The proposed approach, using these innovative tools, differentiates from others in the automated policy refinement technique. Other approaches require that a policy expert either creates policy patterns or creates case-based policies to map high-level policies into low-level policies. Or they require a policy expert to validate the creation of policy refinement traces. It was substantiated that the tools and the model together can hide the complexity and allows non-policy experts to define high-level policies. In addition, these tools ensure that low-level policies are generated correctly according to the adaptive web service's model. These adaptive web service models are described by developers; these are non-policy experts but, are responsible for creating the web services. These integrated tools also facilitate the description of adaptive web services by hiding the complexity of their model as lexical form, thus encouraging its use by the developers of these web services.

6.5 Future Work

This thesis presented an approach for creating discrete policies to manage adaptive composite services, including a set of integrated tools to facilitate the users in accomplishing the required tasks. However, this is only the beginning for the area of adaptive web service management.

The prototype tools allow users to automatically refine high-level policies, generating discrete low-level policies according to the intentions of the specified policies. However, developers need to describe these adaptive behaviours beforehand. It would be even easier to adopt this approach if the descriptions of these adaptive behaviours could be inferred directly from the web service's implementation. It is envisaged that this can be achieved through annotations in the web service's implementation.

In the proposed approach, adaptive behaviours presented for a composite service is the resultant aggregation of the adaptive behaviours pertaining to its constituent services, without any constraints. It is envisaged that this approach can be extended to provide semantic tagging and filtering mechanisms for the adaptive behaviours with the aim of supporting users in specifying semantically correct high level policies.

References

[1] D. Lewis, O. Conlan, D. O'Sullivan, V. Wade, "*Managing adaptive pervasive computing using knowledge-based service integration and rule-based behavior*", IFIP/IEEE Network Operations and Management Symposium, NOMS 2004, Seoul Korea, April 2004

[2] A. Guerrero, V. Villagra, J. Lopez de Vergara,A. Sanchez-Macian, and J. Berrocal, "Ontology-Based Policy Refinement Using SWRL Rules for Management Information Definitions in OWL", Large Scale Management of Distributed Systems, DSOM 2006, LNCS 4269, pp. 227 – 232, 2006.

[3] T Klie, L Wolf, "Automatic Policy Refinement Using OWL-S and Semantic Infrastructure Information", MACE Workshop, ManWeek 2007, San José, CA, USA 2007

[4] Nematbakhsh, Naser; Mardukhi, Farhad; Mohammadkhani, Hasan; "Proposition of a Query Planner for Semantic Web Services", Services Science, Management and Engineering, 2009. SSME '09. IITA International Conference on 11-12 July 2009 Page(s):366 – 369

[5] Seog-Chan Oh; Hyunyoung Kil; Dongwon Lee; Kumara, S.R.T.; "WSBen: A Web Services Discovery and Composition Benchmark", Web Services, 2006. ICWS '06. International Conference on 18-22 Sept. 2006 Page(s):239 - 248

[6] K. Barrett, J Strassner, S. Meer, W. Donnelly, "Determining the Feasibility of Policy Translation" Modelling Autonomic Communications Environments, 2007. MACE 2007.

[7] J. Bruijn, D. Fensel, M. Kifer, J. Kopeck, R. Lara, H. Lausen, A. Polleres, D. Roman, J. Scicluna, I. Toma, "Relationship of WSMO to other relevant technologies", 2005. WWW Page http://www.w3.org/Submission/WSMO-related/

[8] Sloman, M.; Lupu, E. "Security and management policy specification" Network, IEEE Volume 16, Issue 2, March-April 2002 Page(s):10 - 19 IEEE JNL

[9] Sabata, B.; Chatterjee, S.; Davis, M.; Sydir, J.J.; Lawrence, T.F.;" Taxonomy for QoS specifications", Object-Oriented Real-Time Dependable Systems, 1997. Proceedings., Third International Workshop on 5-7 Feb. 1997 Page(s):100 - 107

[10] Jin Yu; Benatallah, B.; Casati, F.; Daniel, F.; "Understanding Mashup Development" Internet Computing, IEEE Volume 12, Issue 5, Sept.-Oct. 2008 Page(s):44 - 52

[11] Sheng, Q.Z.; Benatallah, B.; Maamar, Z.; Ngu, A.H.H.; "Configurable Composition and Adaptive Provisioning of Web Services", Services Computing, IEEE Transactions on Volume 2, Issue 1, Jan.-March 2009 Page(s):34 - 49

[12] Casati, F.; Ilnicki, S.; Li-Jie Jin; Krishnamoorthy, V.; Ming-Chien Shan; "eFlow: a platform for developing and managing composite e-services" Research Challenges, 2000. Proceedings. Academia/Industry Working Conference on 27-29 April 2000 Page(s):341 - 348

[13] Liangzhao Zeng; Benatallah, B.; Ngu, A.H.H.; Dumas, M.; Kalagnanam, J.; Chang, H.; "QoS-aware middleware for Web services composition", Software Engineering, IEEE Transactions on Volume 30, Issue 5, May 2004 Page(s):311 - 327

[14] M.Hanna, A. Buck, R. Smith, "Fuzzy Petri nets to control vision system and robot behaviour under uncertain situations within an FMS cell", Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the Third IEEE on Pages: 1889 - 1894 vol.3

[15] Tan Phan, Jun Han, Jean-Guy Schneider, Tim Ebringer and Tony Rogers , "Policy-Based Service Registration and Discovery" Lecture Notes in Computer Science, On the Move to Meaningful Internet Systems 2007: OTM 2007, Part I, LNCS 4803, pp. 417–426, 2007

[16] Duma, C.; Herzog, A.; Shahmehri, N.; "Privacy in the Semantic Web: What Policy Languages Have to Offer", Policies for Distributed Systems and Networks, 2007. POLICY '07. Eighth IEEE International Workshop on 13-15 June 2007 Page(s):109 - 118

[17] S. Wright, R. Chadha, G. Lapiotis (eds.), "Special Issue on Policy Based Networking". IEEE Network, Vol. 16, No. 2, March, (2002), 8–56

[18] M. Sloman, "Policy Driven Management for distributed Systems." Plenum Press Journal of Network and Systems Management, Vol. 2, No. 4, (1994), 333–360

[19] Tan Phan; Jun Han; Schneider, J.-G.; Ebringer, T.; Rogers, T.; "A Survey of Policy-Based Management Approaches for Service Oriented Systems", Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on 26-28 March 2008 Page(s):392 - 401

[20] TeleManagement Forum, NGOSS Technology Neutral Architecture Release 4.0, January 2004

[21] C.J. Strassner, "Policy-based Network Management, Solutions for the Next Generation", Elsevier, Morgan Kaufmann Publishers 2004. ISBN:1-55860-859-1.

[22] N. Damianou, N. Dulay, E. Lupu, M. Sloman, T. Tonouchi, "Tools for domain-based policy management of distributed systems," Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP 15-19 April 2002 Page(s):203 - 217

[23] E. Lupu, M. Sloman, "Conflicts in policy-based distributed systems management," Software Engineering, IEEE Transactions on Volume 25, Issue 6, Nov.-Dec. 1999 Page(s):852 - 869.

[24] Naseri, M.; Towhidi, A.; "Qos-Aware Automatic Composition of Web Services Using AI Planners", Internet and Web Applications and Services, 2007. ICIW '07. Second International Conference on 13-19 May 2007 Page(s):29 - 29

[25] Casati, F.; Ilnicki, S.; Jin, L.-J.; Shan, M.-C.; "An open, flexible, and configurable system for service composition", Advanced Issues of E-Commerce and Web-Based Information Systems, 2000. WECWIS 2000. Second International Workshop on 8-9 June 2000 Page(s):125 - 132

[26] R. Hamadi and B. Benatallah, "A Petri-Net-Based Model for Web Service Composition," Proc. 14th Australasian Database Conf. Database Technologies, ACM Press, 2003, pp. 191–200

[27] W3C "Extensible Markup Language (XML)", 2004 (Web reference: http://www.w3.org/TR/xml)

[28] W3C "RDF Vocabulary Description Language 1.0: RDF Schema", 2004 (Web reference: http://www.w3.org/TR/rdf-schema/)

[29] W3C "Web Service Modeling Ontology (WSMO)", 2005 (Web reference: http://www.w3.org/Submission/WSMO/)

[30] W3C "Web Service Modeling Language (WSML)" , 2005 (Web reference: http://www.w3.org/Submission/WSML/)

[31] W3C "Web Service Architecture", 2004 (Web reference: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/)

[32] E. Christensen, F. Curbera, G.Meredith, S.Weerawarana, "Web Services Description Language (WSDL) – version 1.1", http://www.w3.org/TR/wsdl

[33] D. Box, et al. Simple Object Access Protocol – Version 1.1, 2004 (Web reference: http://www.w3.org/TR/2000/NOTE-SOAP-20000508/)

[34] Thatte, S., et al.: Business Process Execution Language for Web Services – Version 1.1. OASIS Standard BPELv11-May052003 (May 2003)

[35] World Wide Web Consortium (W3C), Web Ontology Language (OWL), www.w3.org/2004/OWL/

[36] "OWL-S: Semantic markup for web services", The DAML Service Coalition, http://www.daml.org/services/, October 2002

[37] OMG – Unified Moddeling Language 1.5, March 2003.

[38] N. Damiano, Dulay N, Lupu, E, Sloman M, "The Ponder Policy Specification Language", Proc. Policy 2001: Workshop on Policies for Distributed Systems and Networks, Bristol, UK, Jan. 2001, Springer-Verlag LNCS 1995

[39] Twidle, K.; Dulay, N.; Lupu, E.; Sloman, M.;"Ponder2: A Policy System for Autonomous Pervasive Environments", Autonomic and Autonomous Systems, 2009. ICAS '09. Fifth International Conference on 20-25 April 2009 Page(s):330 - 335

[40] L. Kagal, T. Finin, A. Johshi, "A Policy Language for Pervasive Computing Environment". In Proceedings of IEEE Fourth International Workshop on Policy (Policy 2003).

[41] Uszok, A., Bradshaw, J., Jeffers, R., Suri, N., Hayes, P., Breedy, M., Bunch, L., Johnson, M., Kulkari, S., Lott, J.,"KAoS Policy and Domain Services: Toward a Description-Logic Approach to Policy Representation, Deconflictions,and Enforcement", IEEE 4th International Workshop on Policies for Distributed Systems and Networks, June 04 - 06, 2003, Lake Como, Italy, pp 93-98

[42] G. Tonti, J. M. Bradshaw, R. Jeffers, R. Montanari, N. Suri1, A. Uszok "Semantic Web Languages for Policy Representation and Reasoning: A Comparison of KAoS, Rei, and Ponder". Proc. 2nd Intl. Semantic Web Conference, Sanibel Island, Florida, USA, LNCS 2870 (October 2003) 419-437.

[43] Sun: The AWT in 1.0 and 1.1. WWW Page (2008) http://java.sun.com/products/jdk/awt/

[44] Sun: Creating a GUI with JFC/Swing. WWW Page (2008) http://java.sun.com/docs/books/tutorial/uiswing/.

[45] Eviware: SoapUI testing tool. WWW Page (2008) http:// www.soapui.org/.

[46] Hewlett-Packard Development Company: Jena - A Semantic Web Framework for Java. WWW Page (February 2007) http://jena.sourceforge.net.

[47] Jess "Rule Engine for the Java Platform", (Web reference: http://herzberg.ca.sandia.gov/)

[48] Apache Software Foundation: Apache Tomcat. WWW Page (2007) http://tomcat.apache.org/.

[49] Apache Software Foundation: Apache Axis2/Java – Next Generation Web Services. WWW Page (April 2007) http://ws.apache.org/axis2/index.html.

[50] R. Wies, "Using a Classification of Management Policies for Policy Specification and Policy Transformation", Integrated Network Management, IM 1995, IFIP/IEEE International Symposium 1995.

[51] J. Moffett, M. Sloman, "Policy Hierarchies for Distributed Systems Management,", IEEE JSAC Special issue on Network Management, Volume 11, Issue 9, 1993 Page(s):1404-1414.

[52] A. Bandara, E. Lupu, J. Moffett, A. Russo, "A Goal-based Approach to Policy Refinement," Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on 7-9 June 2004 Page(s):229 – 239.

[53] A. Bandara, E. Lupu, A. Russo, N. Dulay, M. Sloman, P. Flegkas, M. Charalambides, G. Pavlou, “Policy Refinement for DiffServ Quality of Service Management,” Integrated Network Management, 2005. IM 2005. 2005 9th IFIP/IEEE International Symposium on Page(s):469 - 482.

[54] R. Darimont, A. van Lamsweerde, “Formal Refinement Patterns for Goal-Driven Requirements Elaboration”, Fourth ACM Symposium on the Foundations of Software Engineering 1996, FSE 1996 pp. 67-95.

[55] R. A. Kowalski, M. Sergot, “A logic-based Calculus of Events”, New Generation Computing, volume 4 pp. 67-95, 1986.

[56] A. Russo, R. Miller, B. Nuseibeh, J. Kramer, “An Adbuctive Aproach for Analysing Event-based Requirements Specifications”, 18th International Conference on Logic Programing 2002, ICLP 2002.

[57] S Beigi,.S Calo,.D Verma, “Policy transformation techniques in policy-based systems management” Policies for Distributed Systems and Networks, POLICY 2004 June 2004 Page(s):13 - 22

[58] S Calo,.D Verma, “A toolkit for policy enablement in autonomic computing”, International Conference on Autonomic Computing 2004, ICAC 2004 Page(s): 270 - 271

[59] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, G. Pavlou, A.L. Lafuente, “Using linear temporal model checking for goal-oriented policy refinement frameworks”, Policies for Distributed Systems and Networks, 2005. Policy 2005. Sixth IEEE International Workshop on 6-8 June 2005 Page(s):181 – 190.

[60] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, G. Pavlou, “A Functional Solution for Goal-Oriented Policy Refinement”, Policies for Distributed Systems and Networks, 2006. Policy 2006. Seventh IEEE International Workshop on Page(s):133 – 144

[61] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, G. Pavlou, “A Distributed Goal-oriented Policy Refinement Environment”, Network Operations and Management Symposium, 2006. NOMS 2006. 10th IEEE/IFIP Page(s): 1 - 4.

[62] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, G. Pavlou, “A Methodological Approach toward the Refinement Problem in Policy-based Management Systems”, Communications Magazine, IEEE Volume: 44 , Issue: 10 Page(s): 60 - 68.

[63] G. Holzmann, “The SPIN Model Checker: Primer Reference Manual”, A. Wesley Publisher 2004. ISBN: 0-321-22862-6.

[64] J.P. de Albuquerque, H. Krumm, P.L. de Geus, "Policy modeling and refinement for network security systems", Policies for Distributed Systems and Networks 2005. Policy 2005. Sixth IEEE International Workshop on 6-8 June 2005 Page(s):24 – 33.

[65] J.P. de Albuquerque, H. Krumm, P.L. de Geus, "Model-based management of security services in complex network environments", Network Operations and Management Symposium, NOMS 2008. IEEE on Page(s): 1031 - 1036.

[66] Linying Su, D. Chadwick, A. Basden, J. Cunningham, "Automated decomposition of access control policies", Policies for Distributed Systems and Networks 2005. Policy 2005. Sixth IEEE International Workshop on 6-8 June 2005.

[67] Linying Su, D. Chadwick, O. Otenko, R. Laborde, "Coordination between distributed PDPs", Policies for Distributed Systems and Networks 2006. Policy 2006. Seventh IEEE International Workshop 2006.

[68] T. Rochaeli, C. Eckert, "Using patterns paradigm to refine workflow policies", Database and Expert System Applications 2007. DESA 2007. 18th IEEE International Workshop 2007.

[69] T. Rochaeli, C. Eckert, "Expertise Knowledge-based Policy Refinement Process", Policies for Distributed Systems and Networks 2007. Policy 2007. Eighth IEEE International Workshop 2007.

[70] M. Casassa Mont, A. Baldwin, C. Goh, "POWER Prototype: Towards Integrated Policy-based Management", Network Operations and Management Symposium, NOMS 2000, IFIP/IEEE International Symposium 2000 pp. 789 – 802.

[71] "State Machines", OMG – Unified Moddeling Language 1.5, March 2003, Pages(s) 2-140 – 2-169

[72] "Activity Graphs", OMG – Unified Moddeling Language 1.5, March 2003, Pages(s) 2-170 – 2-180

[73] "Use Cases", OMG – Unified Moddeling Language 1.5, March 2003, Pages(s) 2-129 – 2-139

[74] K. Carey, V. Wade. "Using Automated Policy Refinement to Manage Adaptive Composite Services." Network Operations and Management Symposium Workshops, 2008. NOMS Workshops IEEE, Salvador Brazil, April 2008.

[75] William F. Clocksin, Christopher S. Mellish,"Programming in Prolog: Using the ISO Standard. Springer", 5th ed., 2003, ISBN 978-3540006787

[76] Sun: "Developer Resources for Java Technology", WWW Page (2008) http://java.sun.com/.

[77] Steve Burbeck, “How to use Model-View-Controller”, WWW Page (2007) http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html.

[78] John S. Uebersax, “Likert Scales: Dispelling the Confusion”, Statistical Methods for Rater Agreement website. 2006 (Web reference: http://www.john-uebersax.com/stat/likert.htm)

[79] Gentleware AG: “Poseidon for UML”, WWW Page (2008) http://www.gentleware.com/.

[80] R. Akkiraju, et al, “Web Services Semantics (WSDL-S) – version 1.0”, 2007 WWWPage http://www.w3.org/Submission/WSDL-S/

[81] J. Whittle, P. Sawyer, N. Bencomo; B. Cheng, “A Language for Self-Adaptive System Requirements”, International Workshop on Service-Oriented Computing: Consequences for Engineering Requirements, 2008. SOCCER '08.

[82] A. Tripathi, “Challenges designing next-generation middleware systems”, Communications of the ACM , Volume 45 Issue 6, June 2002.

[83] V. Koutsonikola, A. Vakali, “A fuzzy bi-clustering approach to correlate web users and pages”, International Journal of Knowledge and Web Intelligence , Volume 1 Issue ½, August 2009.

[84] F.P. Williams, O. Conlan, “Visualizing Narrative Structures and Learning Style Information in Personalized e-Learning Systems”, ICALT 2007: Seventh IEEE International Conference on Advanced Learning Technologies, 2007.

[85] Y. Krishnamurthy, V., D. Karr, C. Rodrigues, D. Schmidt, “Integration of QoS-Enabled Distributed Object Computing Middleware for Developing Next-Generation Distributed Application”, LCTES '01:Proceedings of the ACM SIGPLAN workshop on Languages, compilers and tools for embedded systems, August 2001.

[86] O. Davidyuk, J. Riekki, V. Rautio, J. Sun, “Context-aware middleware for mobile multimedia applications”, MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia, October 2004.

[87] N. Venkatasubramanian, “Safe 'composability' of middleware services”, Communications of the ACM , Volume 45 Issue 6, June 2002.

[88] J. Keeney, V. Cahill, “Chisel: a policy-driven, context-aware, dynamic adaptation framework”, IEEE 4th International Workshop on Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003.

[89] T. Gruber, “Toward Principles for the Design of Ontologies Used for Knowledge Sharing”. International Journal Human-Computer Studies, 43(5-6):907-928, 1995

[90] T. Gruber, “A translation approach to portable ontology specifications”. Knowledge Acquisition, 5:199-220, 1993.

[91] Sybase: "PowerDesigner", WWW Page (2009) http://www.sybase.com/products/modelingdevelopment/powerdesigner/.

[92] IBM: "Rational Software Architect", WWW Page (2009) http://www.ibm.com/software/awdtools/architect/swarchitect/.

[93] Tigris.org: "ArgoUML", WWW Page (2009) http:// http://argouml.tigris.org/.

[94] Umbrello Team: "Umbrello UML Modeller", WWW Page (2009) http://uml.sourceforge.net/.

[95] Stanford University: "Protege", WWW Page (2009) http://protege.stanford.edu/.

[96] University of Maryland: "Pellet", WWW Page (2009) http://pellet.owldl.com/.

[97] OBO Foundry, Open Biomedical Ontologies, WWW Page (2009) http://obofoundry.org/

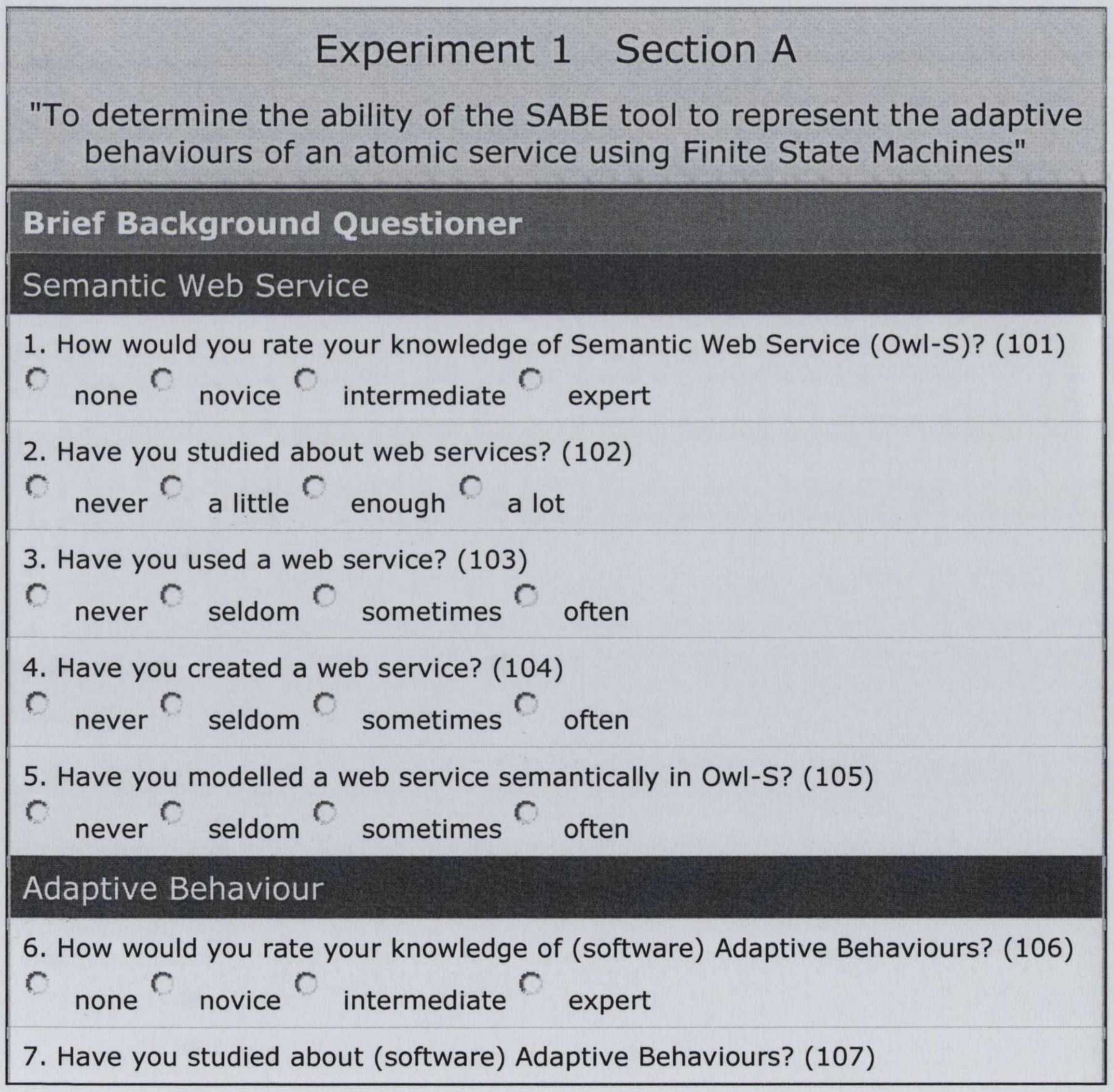
[98] P. Clark and B. Porter., Knowledge Machine, WWW Page (2009) http://www.cs.utexas.edu/users/mfkb/RKF/km.html

Appendix

Appendix A – SABE Tool Usability Evaluation Experiment

The SABE tool usability evaluation presented to the users, which consists of three usability tests. Each usability test presents the instructions, experiment tasks, and the usability questionnaire.

Instructions and Questionnaire for SABE Usability Test 1



Experiment 1 Section A

"To determine the ability of the SABE tool to represent the adaptive behaviours of an atomic service using Finite State Machines"

Brief Background Questioner

Semantic Web Service

1. How would you rate your knowledge of Semantic Web Service (Owl-S)? (101)
○ none ○ novice ○ intermediate ○ expert

2. Have you studied about web services? (102)
○ never ○ a little ○ enough ○ a lot

3. Have you used a web service? (103)
○ never ○ seldom ○ sometimes ○ often

4. Have you created a web service? (104)
○ never ○ seldom ○ sometimes ○ often

5. Have you modelled a web service semantically in Owl-S? (105)
○ never ○ seldom ○ sometimes ○ often

Adaptive Behaviour

6. How would you rate your knowledge of (software) Adaptive Behaviours? (106)
○ none ○ novice ○ intermediate ○ expert

7. Have you studied about (software) Adaptive Behaviours? (107)

○ never ○ a little ○ enough ○ a lot

8. Have you used a piece of software with some adaptive behaviour? (108)
○ never ○ seldom ○ sometimes ○ often

9. Have you configured an adaptive piece of software or application? (109)
○ never ○ seldom ○ sometimes ○ often

10. Have you created a piece of software with some adaptive behaviour? (110)
○ never ○ seldom ○ sometimes ○ often

Finite State Machine

11. How would you rate your knowledge of Finite State Machine? (111)
○ none ○ novice ○ intermediate ○ expert

12. Have you studied about Finite State Machine? (112)
○ never ○ a little ○ enough ○ a lot

13. Have you seen a piece of software described as a Finite State Machine? (113)
○ never ○ seldom ○ sometimes ○ often

14. Have you described a piece of software semantically? (114)
○ never ○ seldom ○ sometimes ○ often

15. Have you described a piece of software using Finite State Machine? (115)
○ never ○ seldom ○ sometimes ○ often

Using Application

16. Have you used an application which uses a tree structure to represent data (such as folders)? (116)
○ never ○ seldom ○ sometimes ○ often

17. Have you used an application which uses a table to display data? (117)
○ never ○ seldom ○ sometimes ○ often

18. Have you used an application which uses a table to allow user to edit values? (118)
○ never ○ seldom ○ sometimes ○ often

Prerequisite Information

Semantic Web Service is a semantic description of a web service.
It uses an ontological language to add semantics to web service descriptions.
It describes services via its inputs, outputs, preconditions, and effects, as well as its composition.

Adaptive Behaviour is a configurable feature or a setting that changes the behaviour of a service.

Finite State Machine is a set of semantic concepts that can be used to model discrete behaviours of any software system. They can be used to specify the behaviour of individual entities or to define the interactions between entities.

Experiment Description

Service Adaptive Behaviour Editor (SABE) is a tool that allows a developer to describe any adaptive behaviour the developer would like to expose for a given service.

SABE tool allows a developer to open a service description described in OWL-S and it displays the service's details.
A service has a process that performs its task which contains a description of its inputs and outputs.

SABE tool allows a user to view the description of any adaptive behaviour belonging to a given service.
Where the adaptive behaviour is modelled as a Finite State Machine (FSM).

SABE tool provides a basic FSM for a service with IdleState, InputState, ProcessState, and OutputState depending on the service.
Adaptive behaviours can then be appended to any of these states as FSM.

Before beginning the experiment, you must first download the Service Adaptive Behaviour Editor (SABE) tool by clicking on sabe.zip.
And you must also download a set of semantic web service descriptions by clicking on owl.zip.

Please read and complete the experiment tasks provided, and to help accomplish these tasks please follow the instructions are provided.

Experiment Tasks

1. Please attempt the tasks bellow and then complete the questionnaire provided. To help accomplishing these tasks, follow the instructions provided.

Material Provided

2. Service Adaptive Behaviour Editor (SABE) tool. Download sabe.zip.

3. Semantic web service descriptions. Download owl.zip.

4. Please follow the instructions provided to run the SABE tool.

Service Provided

5. You are provided with a service ServiceAdapt1.

6. ServiceAdapt1 is an adaptive print service, which prints a given document. This service is adaptable to different printing modes. E.g. print single page, double page.

7. Its inputs are: DocumentName, which informs the service of the document it needs to print;
NumberOfPages, which informs the service of how many pages the document has;
NumberOfColours, which informs the service of how many colours it needs to use, default is 1.

8. Its output is NumberOfSheet, which informs you how many sheets it used for the print job.

9. Your **initial task** is to load this service in the SABE tool.

Adaptive Behaviour

10. This service has 4 adaptive behaviours described for it.

11. It has an adaptive behaviour EconomyMode, which configures the service to print 2 pages per sheet double sided via printmode (2, doublesided)

12. Your **first task** is to analyze how this adaptive behaviour is described in the SABE tool.

13. It has an adaptive behaviour ExpensiveMode, which configures the service to print 1 page per sheet single sided via printmode (1, singlesided)

14. Your **second task** is to analyze how this adaptive behaviour is described in the SABE tool and answer the questions on the web site.

15. It has an adaptive behaviour ColourPrinting, which configures the service to print in colour by first checking if uses a colour cartage (checkcartage(type_colour)) and if colourcartage is present when Printing (Event) then set the printer to use colour cartage (usecartage(type_colour))

16. Your **third task** is to analyze how this adaptive behaviour is described in the SABE tool and answer the questions on the web site.

Experiment Instructions

Material Provided

1. Service Adaptive Behaviour Editor (SABE) tool. Download sabe.zip.

2. Semantic web service descriptions. Download owl.zip.

Steps for Installation

3. Make sure you have Java 1.4.x or higher installed in your computer (from dos prompt type java -version).

4. Download and unzip sabe.zip to c:\sabe.

5. Download and unzip owl.zip to c:\owl.

Steps for Running the Application

6. Run c:\sabe\sabe.bat.

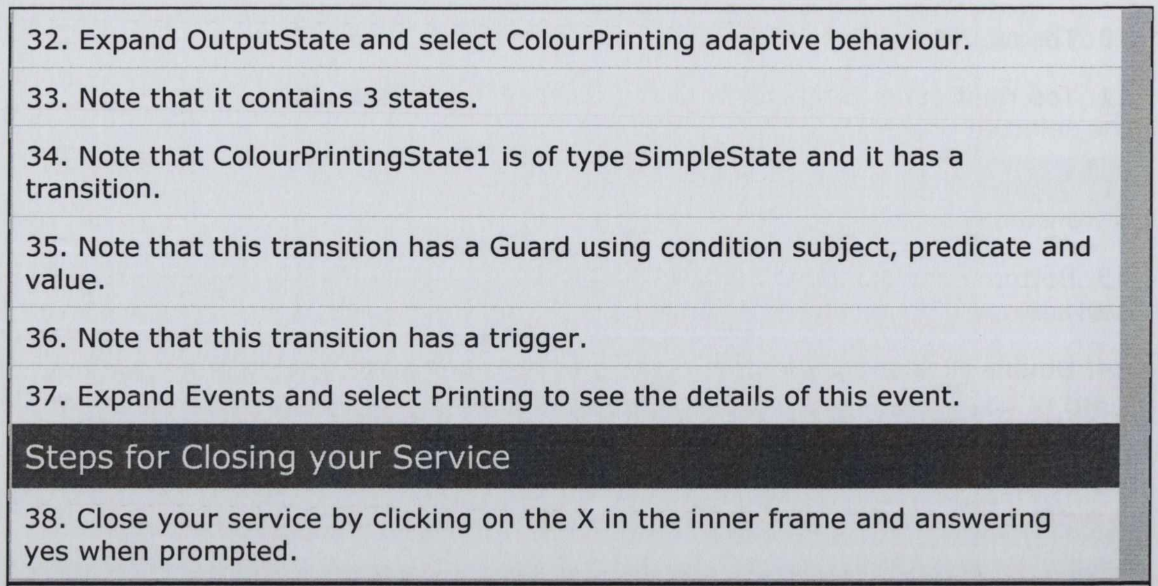
7. From the SABE application open c:\owl\ServiceAdapt1.owl.

Steps for Operating SABE

8. Top left panel details of the service and its composition.

9. Bottom left panel contains details of the selected process such as its parameters (Inputs, Outputs) and Finite State Machine.

10. The service is supported by a process.
11. Top right panel contains the details of the Finite State Machine belonging to the selected process.
12. Bottom right panel contains the details of the selected node (e.g. a state or a transition) of the Finite State Machine.
13. Bottom panel contains the details of the services dataflow for a composite service.
14. Double click on a node or click the plus sign to expand a node of a tree.
15. Hover over for information on the node.
16. Right click on a node to perform action if any.
Steps for Browsing EconomyMode Adaptive Behaviour
17. Expand the service node on the left panel, a process will appear under it.
18. Select the process, its description will appear in the panel bellow and FSM on the top right panel.
19. Expand the FSM node on the top right panel.
20. Navigate through the different states (e.g. InputState) and select EconomyMode adaptive behaviour.
21. Its name and description will be shown in the bottom right panel.
22. Hover over the description to see a tool tip with the complete description.
23. Expand the adaptive behaviour (EconomyMode) node to see its states.
24. There should be an EconomyModeInitialState (PseudoState) and an EconomyModeState1 (FinalState).
25. Select the state to see its name, its type and its activity (Note that a pseudo state has a kind instead of activity).
26. An initial state must always have a transition with a target pointing to the first state that supports this adaptive behaviour.
27. A final state cannot have any transition, it is the last state of the adaptive behaviour.
28. Simple and final states have activities that can be set.
29. Expand the initial state and select its transition.
30. This transition has its target pointing at next state (EconomyModeState1).
Steps for Browsing ExpensiveMode Adaptive Behaviour
31. Repeat the same procedure for viewing the ExpensiveMode adaptive behaviour.
Steps for Browsing ColourPrinting Adaptive Behaviour



32. Expand OutputState and select ColourPrinting adaptive behaviour.

33. Note that it contains 3 states.

34. Note that ColourPrintingState1 is of type SimpleState and it has a transition.

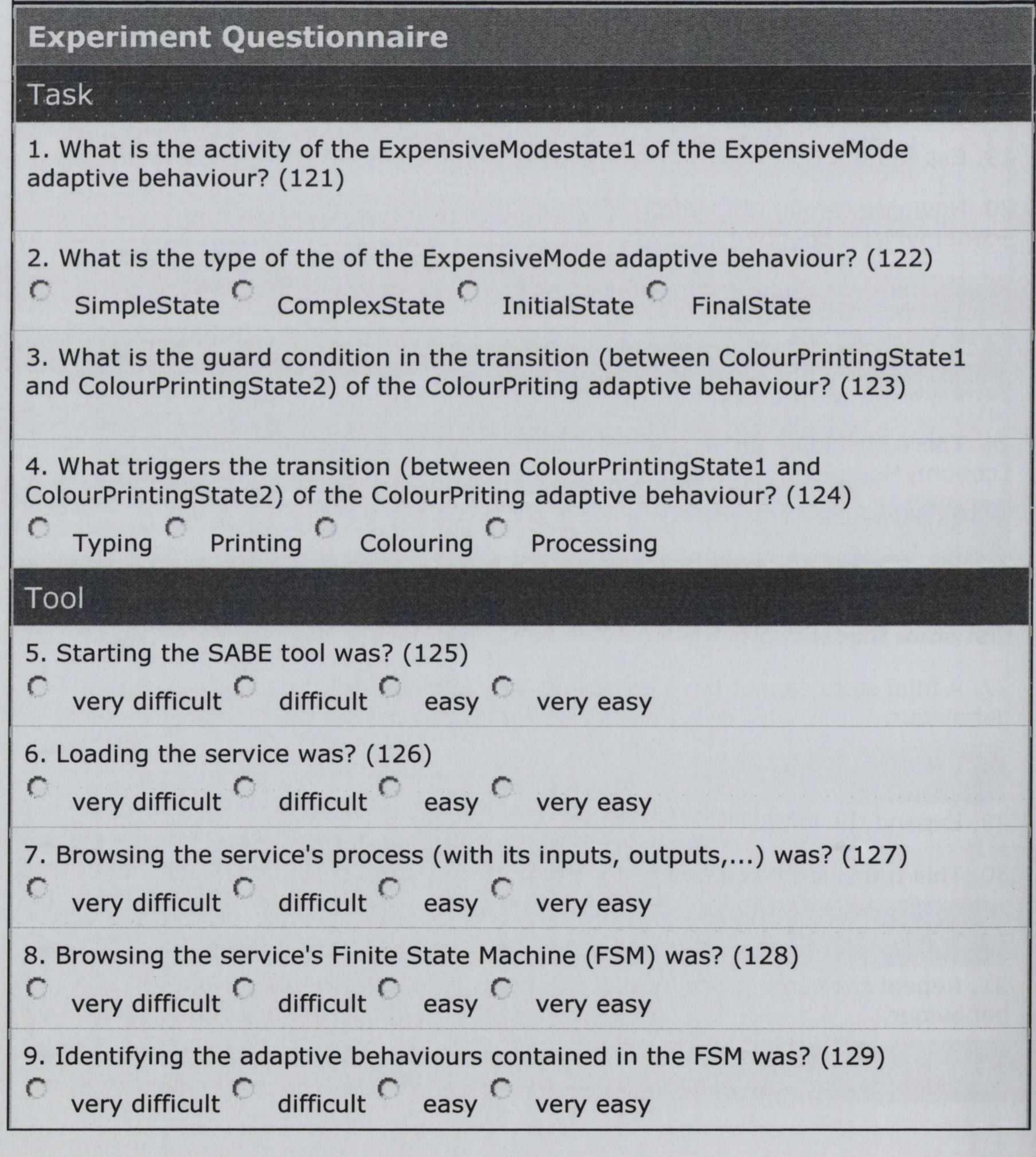
35. Note that this transition has a Guard using condition subject, predicate and value.

36. Note that this transition has a trigger.

37. Expand Events and select Printing to see the details of this event.

Steps for Closing your Service

38. Close your service by clicking on the X in the inner frame and answering yes when prompted.



Experiment Questionnaire

Task

1. What is the activity of the ExpensiveModestate1 of the ExpensiveMode adaptive behaviour? (121)

2. What is the type of the of the ExpensiveMode adaptive behaviour? (122)
○ SimpleState ○ ComplexState ○ InitialState ○ FinalState

3. What is the guard condition in the transition (between ColourPrintingState1 and ColourPrintingState2) of the ColourPriting adaptive behaviour? (123)

4. What triggers the transition (between ColourPrintingState1 and ColourPrintingState2) of the ColourPriting adaptive behaviour? (124)
○ Typing ○ Printing ○ Colouring ○ Processing

Tool

5. Starting the SABE tool was? (125)
○ very difficult ○ difficult ○ easy ○ very easy

6. Loading the service was? (126)
○ very difficult ○ difficult ○ easy ○ very easy

7. Browsing the service's process (with its inputs, outputs,...) was? (127)
○ very difficult ○ difficult ○ easy ○ very easy

8. Browsing the service's Finite State Machine (FSM) was? (128)
○ very difficult ○ difficult ○ easy ○ very easy

9. Identifying the adaptive behaviours contained in the FSM was? (129)
○ very difficult ○ difficult ○ easy ○ very easy

10. Viewing the details of an adaptive behaviour was? (130)

○ very difficult ○ difficult ○ easy ○ very easy

11. Identifying the actions performed by an adaptive behaviour was? (131)

○ very difficult ○ difficult ○ easy ○ very easy

12. Identifying the sequence of actions performed by an adaptive behaviour was? (132)

○ very difficult ○ difficult ○ easy ○ very easy

13. Sabe tool made the process of browsing the description of an adaptive behaviour? (133)

○ very difficult ○ difficult ○ easy ○ very easy

14. Sabe tool made the process of identifying the actions of an adaptive behaviour? (134)

○ very difficult ○ difficult ○ easy ○ very easy

15. Sabe tool made the process of identifying the sequence of actions of an adaptive behaviour? (135)

○ very difficult ○ difficult ○ easy ○ very easy

16. To understand the description of adaptive behaviours using SABE tool was? (136)

○ very confusing ○ confusing ○ clear ○ very clear

17. To understand the description of states describing an action of adaptive behaviours using SABE tool was? (137)

○ very confusing ○ confusing ○ clear ○ very clear

18. To understand the description of transitions describing a sequence of actions of adaptive behaviours using SABE tool was? (138)

○ very confusing ○ confusing ○ clear ○ very clear

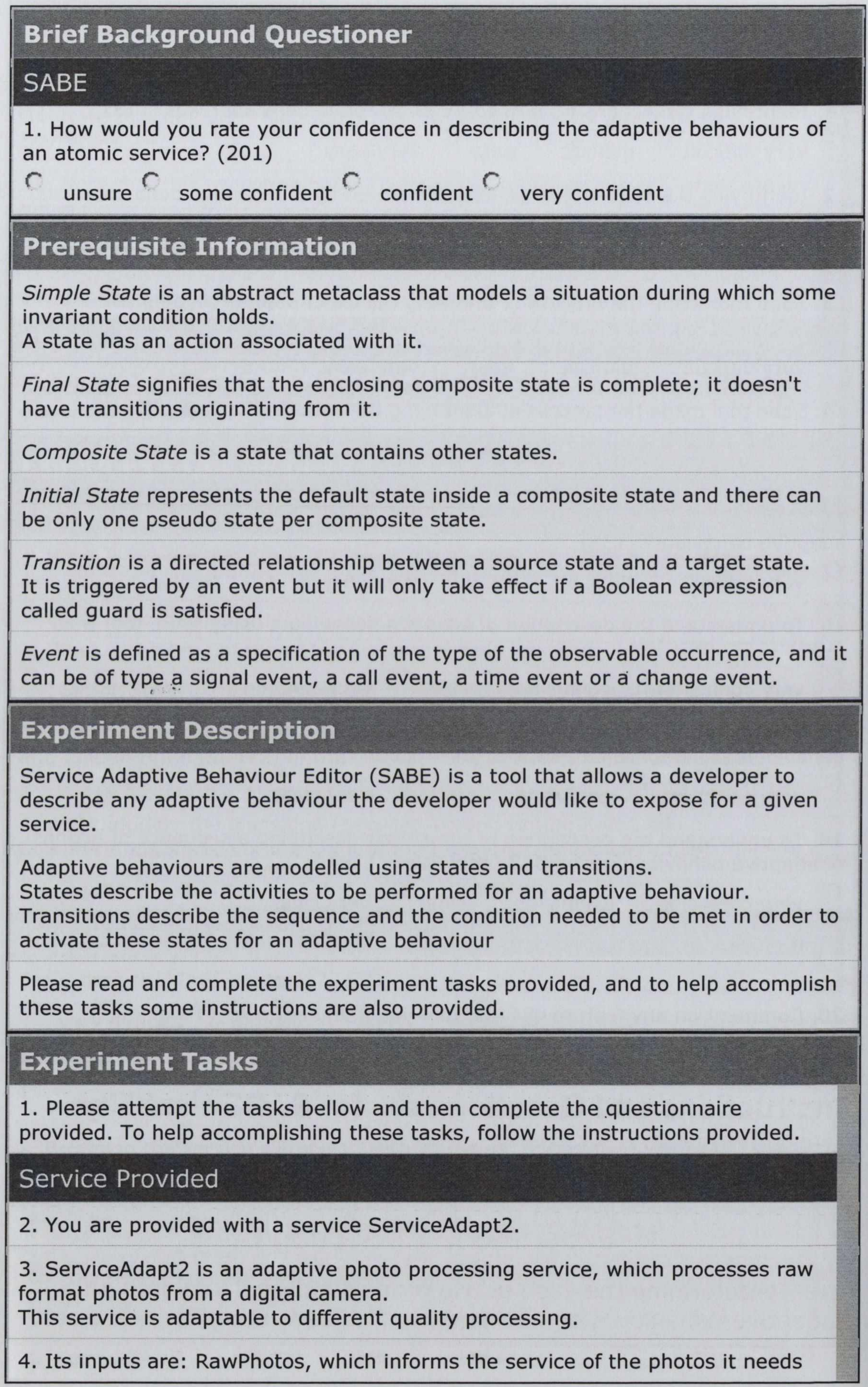
19. Comment on any feature of SABE tool you found useful? (139)

20. Comment on any feature of SABE tool you found difficult? (140)

Instructions and Questionnaire for SABE Usability Test 2

Experiment 1 Section B

"To determine the ease of use of the SABE tool in defining the adaptive behaviours of an atomic service as Finite State Machines."



Brief Background Questioner

SABE

1. How would you rate your confidence in describing the adaptive behaviours of an atomic service? (201)

○ unsure ○ some confident ○ confident ○ very confident

Prerequisite Information

Simple State is an abstract metaclass that models a situation during which some invariant condition holds.
A state has an action associated with it.

Final State signifies that the enclosing composite state is complete; it doesn't have transitions originating from it.

Composite State is a state that contains other states.

Initial State represents the default state inside a composite state and there can be only one pseudo state per composite state.

Transition is a directed relationship between a source state and a target state. It is triggered by an event but it will only take effect if a Boolean expression called guard is satisfied.

Event is defined as a specification of the type of the observable occurrence, and it can be of type a signal event, a call event, a time event or a change event.

Experiment Description

Service Adaptive Behaviour Editor (SABE) is a tool that allows a developer to describe any adaptive behaviour the developer would like to expose for a given service.

Adaptive behaviours are modelled using states and transitions.
States describe the activities to be performed for an adaptive behaviour.
Transitions describe the sequence and the condition needed to be met in order to activate these states for an adaptive behaviour

Please read and complete the experiment tasks provided, and to help accomplish these tasks some instructions are also provided.

Experiment Tasks

1. Please attempt the tasks bellow and then complete the questionnaire provided. To help accomplishing these tasks, follow the instructions provided.

Service Provided

2. You are provided with a service ServiceAdapt2.

3. ServiceAdapt2 is an adaptive photo processing service, which processes raw format photos from a digital camera.
This service is adaptable to different quality processing.

4. Its inputs are: RawPhotos, which informs the service of the photos it needs

to process;
PhotoCategory, which informs the service of what type of photos;
PhotoSize, which informs the service of what size the photos have to be.

5. Its output is ProcessedPhotos, which returns the processed photos.

6. The default behaviour of the service is to process photos in colour and use low resolution.

7. Your **initial task** is to load this service in the SABE tool and create a FSM for it (which will contain the adaptive behaviours).

Generating Finite State Machine

8. This service needs a Finite State Machine to contain the adaptive behaviours

9. Your **first task** is to generate a Finite State Machine for this service's process.

Creating Adaptive Behaviours

10. This service has an adaptive behaviour HighPhotoQuality, which configures the service to process the photos using high resolution (setResolution(high))

11. Your **second task** is to model this adaptive behaviour using the SABE tool according to its description.

12. This service has an adaptive behaviour BlackWhitePhoto, which configures the service to process the photos in black&white (setColour(false))

13. Your **third task** is to model this adaptive behaviour using the SABE tool according to its description.

14. This service has an adaptive behaviour RemoveRedEye, which removes red eyes in the photos by first finding a red eye in the photo (searchRedEye()) and if redeyelocation is not zero when ProcessingPhoto (Event) then remove it (removeRedEye())

15. Your **fourth task** is to model this adaptive behaviour using the SABE tool according to its description.

Saving Your Work

16. Now that you have finished describing these adaptive behaviours, your **final task** is to save them by clicking on File and then save.

Experiment Instructions

Material Provided

1. Service Adaptive Behaviour Editor (SABE) tool. Download sabe.zip.

2. Semantic web service descriptions. Download owl.zip.

Steps for Running the Application

3. Run c:\sabe\sabe.bat.

4. From the SABE application open c:\owl\ServiceAdapt2.owl.

Steps for Generating a Finite State Machine

5. Expand the service node on the left panel, a process will appear under it.

6. Select the process, its description will appear in the panel bellow.

7. Select the StateMachine node from the process detail panel.

8. Right click and add Finite State Machine.

9. A FSM is generated and its details can be seen on the top right panel.

Steps for Adding HighPhotoQuality Adaptive Behaviour

10. Expand the FSM tree on the top right panel.

11. Select ProcessState and right click to add an adaptive behaviour.

12. Give the adaptive behaviour a proper name (HighPhotoQuality) and description in the bottom right panel.

13. Right click the newly added adaptive behaviour to add a state.

14. The first state is an Initial State (PseudoState) which must have a transition with a target to another state.

15. Right click the newly added adaptive behaviour again to add another state.

16. Select the added state and name it with any appropriate name you desire (HighPhotoQualityState1).

17. Change the state type to FinalState (the last state - this state does not have transitions).

18. Set the appropriate activity for this state (setResolution(high)).

19. Right click the initial state and add a transition.

20. Select the added transition and select the desired state (HighPhotoQualityState1) as its target (Press ENTER (keyboard button) after doing this action to assure your selection).

Steps for Adding BlackWhitePhoto Adaptive Behaviour

21. Repeat the same procedure for adding the BlackWhitePhoto Adaptive Behaviour.

22. Use the appropriate names and activity (mentioned in the task section) for adding BlackWhitePhoto Adaptive Behaviour.

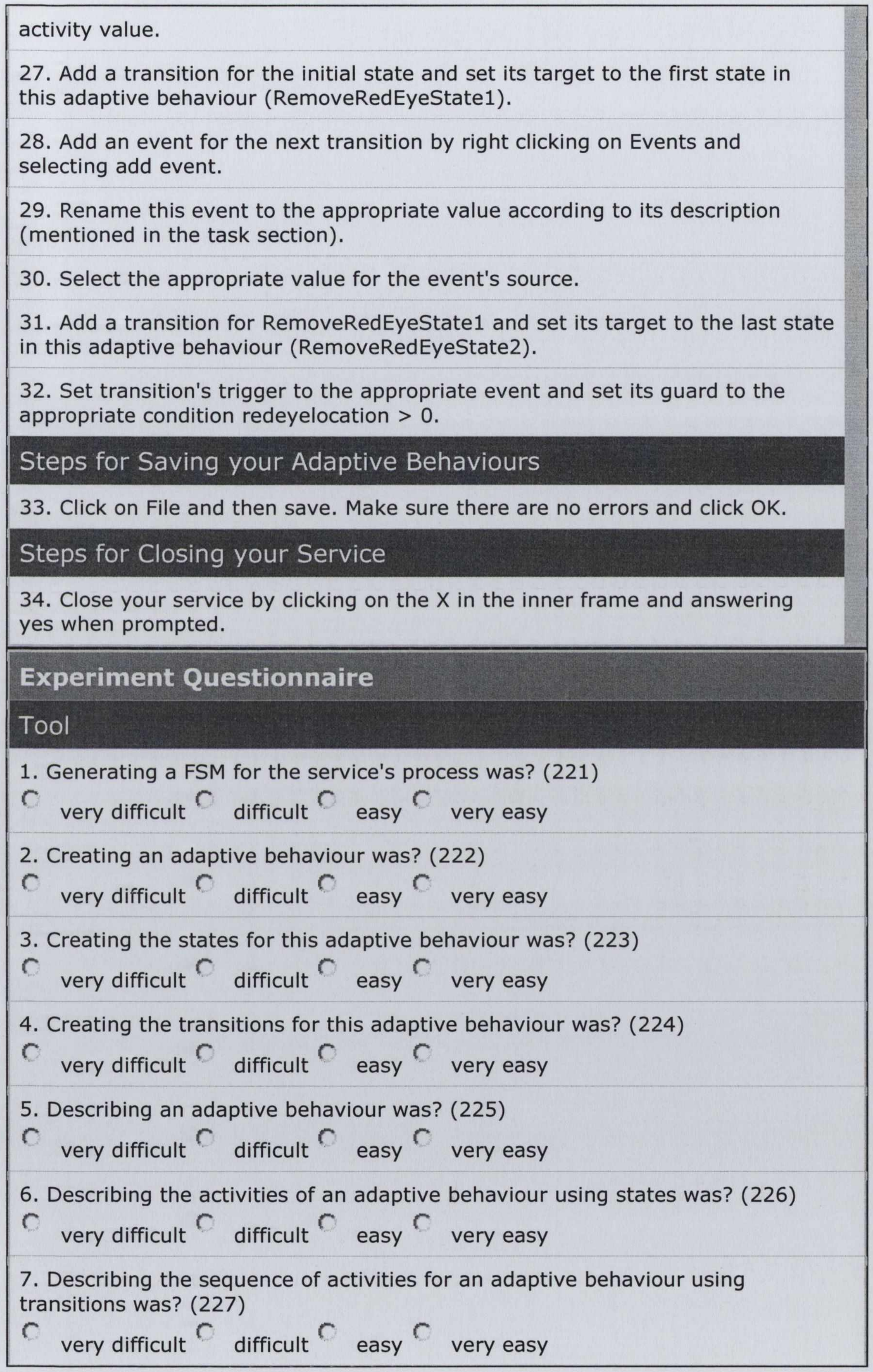
Steps for Adding RemoveRedEye Adaptive Behaviour

23. Add RemoveRedEye adaptive behaviour under InputState.

24. Add an initial state.

25. Add a simplestate state (RemoveRedEyeState1) with the appropriate activity.

26. Add a FinalState state (RemoveRedEyeState2) and set the appropriate



activity value.

27. Add a transition for the initial state and set its target to the first state in this adaptive behaviour (RemoveRedEyeState1).

28. Add an event for the next transition by right clicking on Events and selecting add event.

29. Rename this event to the appropriate value according to its description (mentioned in the task section).

30. Select the appropriate value for the event's source.

31. Add a transition for RemoveRedEyeState1 and set its target to the last state in this adaptive behaviour (RemoveRedEyeState2).

32. Set transition's trigger to the appropriate event and set its guard to the appropriate condition redeyelocation > 0.

Steps for Saving your Adaptive Behaviours

33. Click on File and then save. Make sure there are no errors and click OK.

Steps for Closing your Service

34. Close your service by clicking on the X in the inner frame and answering yes when prompted.

Experiment Questionnaire

Tool

1. Generating a FSM for the service's process was? (221)
○ very difficult ○ difficult ○ easy ○ very easy

2. Creating an adaptive behaviour was? (222)
○ very difficult ○ difficult ○ easy ○ very easy

3. Creating the states for this adaptive behaviour was? (223)
○ very difficult ○ difficult ○ easy ○ very easy

4. Creating the transitions for this adaptive behaviour was? (224)
○ very difficult ○ difficult ○ easy ○ very easy

5. Describing an adaptive behaviour was? (225)
○ very difficult ○ difficult ○ easy ○ very easy

6. Describing the activities of an adaptive behaviour using states was? (226)
○ very difficult ○ difficult ○ easy ○ very easy

7. Describing the sequence of activities for an adaptive behaviour using transitions was? (227)
○ very difficult ○ difficult ○ easy ○ very easy

8. Sabe tool made the process of generating a FSM for the service? (228)

○ very difficult ○ difficult ○ easy ○ very easy

9. Sabe tool made the process of describing an adaptive behaviour? (229)

○ very difficult ○ difficult ○ easy ○ very easy

10. Sabe tool made the process of describing the activities of an adaptive behaviour? (230)

○ very difficult ○ difficult ○ easy ○ very easy

11. Sabe tool made the process of describing the sequence of activities for an adaptive behaviour? (231)

○ very difficult ○ difficult ○ easy ○ very easy

12. To understand the process of generating a FSM was? (232)

○ very confusing ○ confusing ○ clear ○ very clear

13. To understand the process of describing an adaptive behaviour was? (233)

○ very confusing ○ confusing ○ clear ○ very clear

14. To understand the process of describing the activities of an adaptive behaviour as states was? (234)

○ very confusing ○ confusing ○ clear ○ very clear

15. To understand the process of describing the sequence of activities for an adaptive behaviour as transitions was? (235)

○ very confusing ○ confusing ○ clear ○ very clear

16. Comment on any feature of SABE tool you found useful? (236)

17. Comment on any feature of SABE tool you found difficult? (237)

Instructions and Questionnaire for SABE Usability Test 3

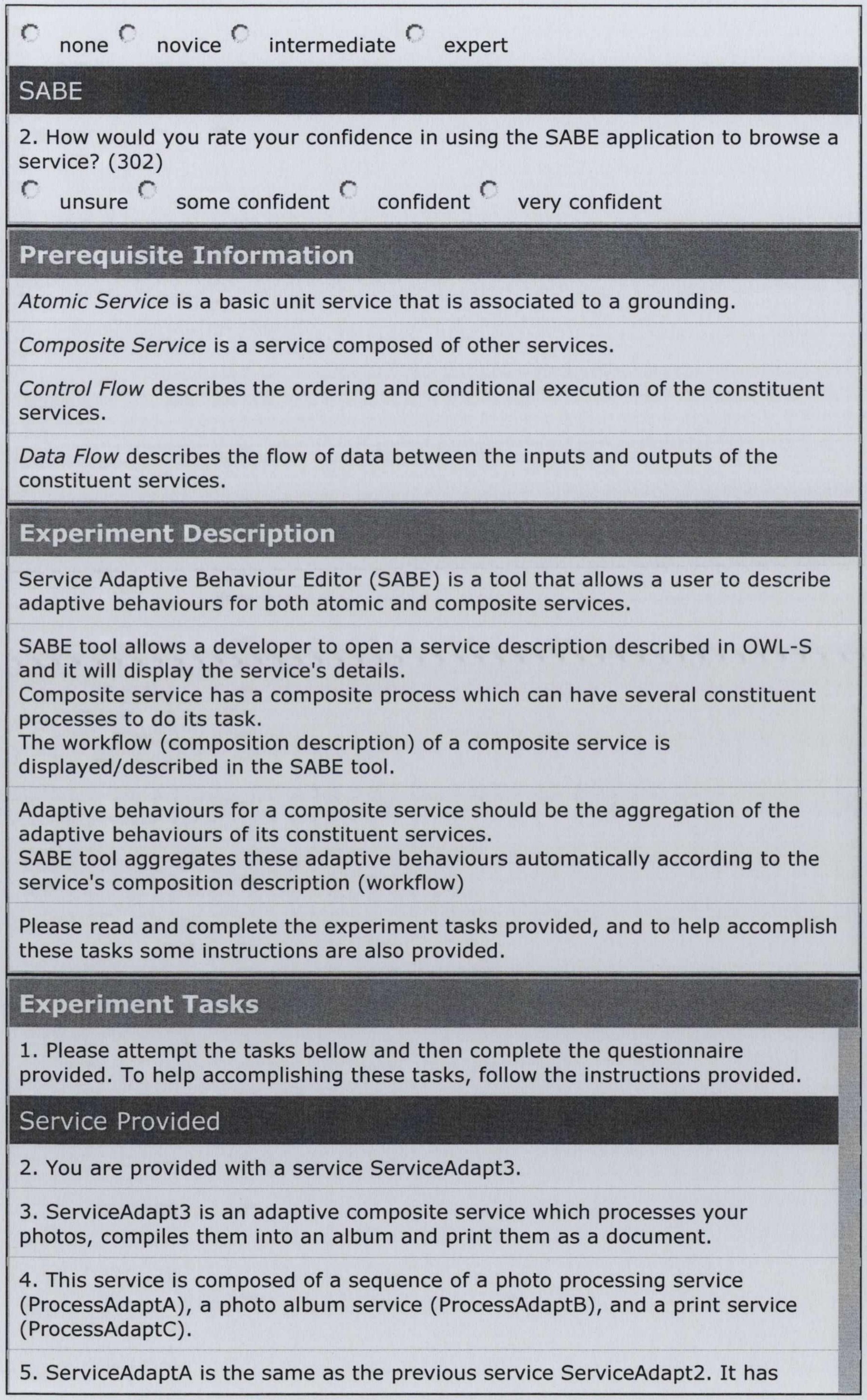
Experiment 1 Section C

"To determine the ease of use and ease of comprehension of the SABE tool in representing the resultant automatic aggregated Finite State Machine describing the adaptive behaviours of a composite service."

Brief Background Questioner

Semantic Web Service

1. How would you rate your understand of service composition? (301)



○ none ○ novice ○ intermediate ○ expert

SABE

2. How would you rate your confidence in using the SABE application to browse a service? (302)

○ unsure ○ some confident ○ confident ○ very confident

Prerequisite Information

Atomic Service is a basic unit service that is associated to a grounding.

Composite Service is a service composed of other services.

Control Flow describes the ordering and conditional execution of the constituent services.

Data Flow describes the flow of data between the inputs and outputs of the constituent services.

Experiment Description

Service Adaptive Behaviour Editor (SABE) is a tool that allows a user to describe adaptive behaviours for both atomic and composite services.

SABE tool allows a developer to open a service description described in OWL-S and it will display the service's details.
Composite service has a composite process which can have several constituent processes to do its task.
The workflow (composition description) of a composite service is displayed/described in the SABE tool.

Adaptive behaviours for a composite service should be the aggregation of the adaptive behaviours of its constituent services.
SABE tool aggregates these adaptive behaviours automatically according to the service's composition description (workflow)

Please read and complete the experiment tasks provided, and to help accomplish these tasks some instructions are also provided.

Experiment Tasks

1. Please attempt the tasks bellow and then complete the questionnaire provided. To help accomplishing these tasks, follow the instructions provided.

Service Provided

2. You are provided with a service ServiceAdapt3.

3. ServiceAdapt3 is an adaptive composite service which processes your photos, compiles them into an album and print them as a document.

4. This service is composed of a sequence of a photo processing service (ProcessAdaptA), a photo album service (ProcessAdaptB), and a print service (ProcessAdaptC).

5. ServiceAdaptA is the same as the previous service ServiceAdapt2. It has

adaptive behaviours HighPhotoQuality, BlackWhitePhoto, and RemoveRedEye.

6. ServiceAdaptC is the same as the previous service ServiceAdapt1. It has adaptive behaviours EconomyMode, IntermediateMode, ExpensiveMode, and ColourPriting.

7. ServiceAdaptB is an adaptive service for creating photo albums from given photos in standard formats. Its default behaviour is to create a photo album.

8. ServiceAdaptB has an adaptive behaviour CreateCalendar, which creates a calendar with the first 12 photos (createCalendar()).

9. ServiceAdaptB has another adaptive behaviour CreatePostcard, which creates an album of postcards with the given photos (createPostcard()).

10. Your **initial task** is to load this service in the SABE tool.

Aggregating Adaptive Behaviours

11. Please note that each of the constituent services contains at least one adaptive behaviour described using FSM.

12. The adaptive behaviours of a composite service is the aggregation of the adaptive behaviours of its constituent services.

13. Your **first task** is to aggregate the adaptive behaviours of the constituent services for the composite service ServiceAdapt3.

Resultant Aggregated Adaptive Behaviours

14. The adaptive behaviours of a composite service should be displayed in the same manner as of an atomic service.

15. Your **second task** is to analyze the resultant aggregated adaptive behaviours of the composite service ServiceAdapt3 and answer the questions on the web site.

Saving Your Work

16. Now that you have finished aggregating these adaptive behaviours, your **final task** is to save them by clicking on File and then save.

Experiment Instructions

Material Provided

1. Service Adaptive Behaviour Editor (SABE) tool. Download sabe.zip.

2. Semantic web service descriptions. Download owl.zip.

Steps for Running the Application

3. Run c:\sabe\sabe.bat.

4. From the SABE application open c:\owl\ServiceAdapt3.owl.

Steps for Browsing Adaptive Behaviours of the Constituent Services

5. Expand the service node on the left panel, a composite process will appear under it (ProcessAdapt3).

6. Expand this composite process to see that it is composed of a sequence of ProcessAdaptA, ProcessAdaptB, and ProcessAdaptC.

7. Browse through the constituent processes of this composite process. ProcessAdaptA is a photo processing service, ProcessAdaptB is a photo album service, and ProcessAdaptC is a printing service

8. For each of the constituent processes, browse through its FSM on the right panel to familiarize with the adaptive behaviours.

9. ProcessAdaptA has HighPhotoQuality, BlackWhitePhoto, and RemoveRedEye adaptive behaviours.

10. ProcessAdaptB has CreateCalendar, and CreatePostCard adaptive behaviours.

11. ProcessAdaptC has EconomyMode, ExpensiveMode, IntermediateMode, and ColourPrinting adaptive behaviours.

Steps for Aggregating Adaptive Behaviours for a Composite Service

12. Select the topmost composite process i.e. select process ProcessAdapt3 from top left panel, its description will appear in the panel bellow.

13. Expand process ProcessAdapt3 node from bottom left panel.

14. Select the StateMachine and right click, then select add Finite State Machine which will generate a populated FSM on the top right panel.

Steps for Browsing Adaptive Behaviours of a Composite Service

15. Expand the FSM ProcessAdapt3FSM node and browse through the automatically aggregated FSM.

16. Verify that all the adaptive behaviours seen in the constituent services are present in the composite service's FSM.

17. InputState should have an adaptive behaviour called RemoveRedEye.

18. OuputState should have an adaptive behaviour called ColourPrinting.

19. ProcessState should have all other adaptive behaviours.

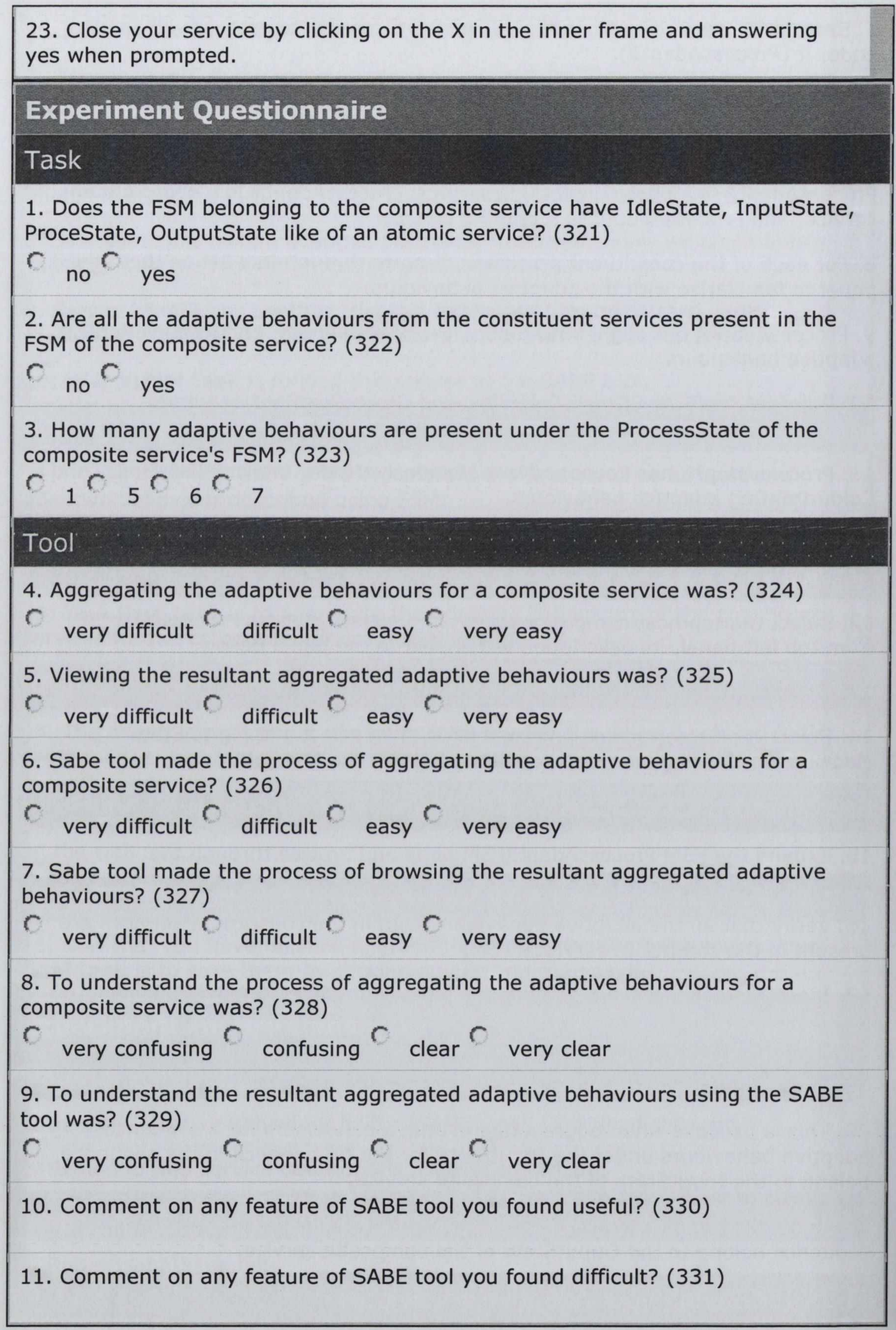
20. This is because when aggregating adaptive behaviours for a sequence, adaptive behaviours under the InputState for the first service in the sequence belong to the InputState of the composite service.

21. And adaptive behaviours under the OutputState for the last service in the sequence belong to the OuputState of the composite service.

Steps for Saving your Adaptive Behaviours

22. Click on File and then save. Make sure there are no errors.

Steps for Closing your Service



23. Close your service by clicking on the X in the inner frame and answering yes when prompted.

Experiment Questionnaire

Task

1. Does the FSM belonging to the composite service have IdleState, InputState, ProceState, OutputState like of an atomic service? (321)

○ no ○ yes

2. Are all the adaptive behaviours from the constituent services present in the FSM of the composite service? (322)

○ no ○ yes

3. How many adaptive behaviours are present under the ProcessState of the composite service's FSM? (323)

○ 1 ○ 5 ○ 6 ○ 7

Tool

4. Aggregating the adaptive behaviours for a composite service was? (324)

○ very difficult ○ difficult ○ easy ○ very easy

5. Viewing the resultant aggregated adaptive behaviours was? (325)

○ very difficult ○ difficult ○ easy ○ very easy

6. Sabe tool made the process of aggregating the adaptive behaviours for a composite service? (326)

○ very difficult ○ difficult ○ easy ○ very easy

7. Sabe tool made the process of browsing the resultant aggregated adaptive behaviours? (327)

○ very difficult ○ difficult ○ easy ○ very easy

8. To understand the process of aggregating the adaptive behaviours for a composite service was? (328)

○ very confusing ○ confusing ○ clear ○ very clear

9. To understand the resultant aggregated adaptive behaviours using the SABE tool was? (329)

○ very confusing ○ confusing ○ clear ○ very clear

10. Comment on any feature of SABE tool you found useful? (330)

11. Comment on any feature of SABE tool you found difficult? (331)

Appendix B – SABE Tool Usability Evaluation Results

The results of the pre-test and usability questionnaires for the three SABE tool's usability evaluation cycles. Analytical results of the Chi-square test performed over the three sets of usability questionnaire results.

Results of the SABE Tool Usability Evaluation Cycle 1

Pre-Test

Users	A	B	C	D	E	F	Z[9]
101. How would you rate your knowledge of Semantic Web Service (Owl-S)?	3	2	4	3	1	4	3
102. Have you studied or read about web services?	3	2	4	4	3	4	3
103. Have you ever used a web service?	4	2	4	4	1	3	3
104. Have you ever created a web service?	2	1	3	4	1	2	2
105. Have you ever described a web service semantically (Owl-S)?	1	1	3	3	1	2	2
106. How would you rate your knowledge of (software) Adaptive Behaviours?	3	3	3	3	2	4	3
107. Have you studied or read about (software) Adaptive Behaviours?	2	3	3	3	2	4	3
108. Have you ever used a piece of software with some adaptive behaviour?	3	2	3	3	2	3	3
109. Have you ever configured an adaptive piece of software or application?	2	1	3	3	1	3	2
110. Have you ever created a piece of software with some adaptive behaviour?	2	1	3	3	1	2	2
111. How would you rate your knowledge of Finite State Machine?	3	3	3	3	3	4	3
112. Have you studied or read about Finite State Machine?	3	3	3	3	4	4	3
113. Have you ever seen a piece of software described as a Finite State Machine?	2	2	3	4	3	4	3

[9] This is the (rounded) average of the pre-test results, where 4 is very positive and 1 is very negative

114. Have you ever described a piece of software semantically?	2	1	3	3	1	3	2
115. Have you ever described a piece of software using Finite State Machine?	2	2	2	3	2	2	2
301. How would you rate your understanding of service composition?	3	2	4	3	2	4	3
302. How would you rate your confidence in using the SABE application to browse a service?	2	2	3	2	2	2	2

Results of the SABE Tool Usability Evaluation Cycle 1

Pre-Test

Users	A	B	C	D	E	F	Z[10]
121. What is the activity of the ExpensiveModestatel of the ExpensiveMode adaptive behaviour?	3	2	4	3	1	4	3
122. What is the type of the of the ExpensiveMode adaptive behaviour?	2	1	3	4	1	2	2
123. What is the guard condition in the transition (between ColourPrintingStatel and ColourPrintingState2) of the ColourPriting adaptive behaviour?	1	1	3	3	1	2	2
124. What triggers the transition (between ColourPrintingStatel and ColourPrintingState2) of the ColourPriting adaptive behaviour?	3	3	3	3	2	4	3
321. Does the FSM belonging to the composite service have IdleState, InputState, ProceState, OutputState like of an atomice service?	2	3	3	3	2	4	3
322. Are all the adaptive behaviours from the constituent services present in the FSM of the composite service?	3	2	3	3	2	3	3
323. How many adaptive behaviours are present under the ProcessState of the composite service's FSM?	2	1	3	3	1	3	2
	2	1	3	3	1	2	2
	3	3	3	3	3	4	3
	3	3	3	3	4	4	3
	2	2	3	4	3	4	3

[10] This is the (rounded) average of the pre-test results, where 4 is very positive and 1 is very negative

Results of the SABE Tool Usability Evaluation Cycle 1 Questionnaire

Users	A	B	C	D	E	F
121. Starting the SABE tool was?	4	4	4	3	3	4
122. Loading the service was?	4	4	4	3	4	4
123. Browsing the service's process (with its inputs, outputs,...) was?	3	4	4	3	3	3
124. Browsing the service's Finite State Machine (FSM) was?	3	4	3	3	3	3
125. Identifying the adaptive behaviours contained in the FSM was?	3	4	3	2	2	3
126. Viewing the details of an adaptive behaviour was?	3	4	3	3	3	2
127. Identifying the actions performed by an adaptive behaviour was?	3	4	3	2	3	2
128. Identifying the sequence of actions performed by an adaptive behaviour was?	3	4	3	2	3	1
129. To understand the description of adaptive behaviours was?	3	4	3	2	3	2
130. To understand the description of states describing an action of adaptive behaviours was?	3	4	3	2	3	3
131. To understand the description of transitions describing a sequence of actions of adaptive behaviours was?	3	4	3	3	3	2
132. Sabe tool made the process of browsing the description of an adaptive behaviour?	3	4	3	3	3	3
133. Sabe tool made the process of identifying the actions of an adaptive behaviour?	3	4	3	2	3	3
134. Sabe tool made the process of identifying the sequence of actions of an adaptive behaviour?	3	4	3	3	3	2
135. When browsing adaptive behaviours, would you rate SABE tool as?	3	4	4	3	3	3
221. Generating a FSM for the service's process was?	3	4	4	3	3	3
222. Creating an adaptive behaviour was?	3	4	4	2	2	3
223. Creating the states for this adaptive behaviour was?	3	4	3	3	3	3
224. Creating the transitions for this adaptive behaviour was?	3	4	3	3	2	4
225. Describing an adaptive behaviour was?	3	4	3	3	3	3
226. Describing the activities of an adaptive behaviour using states was?	3	4	2	3	3	3

227. Describing the sequence of activities for an adaptive behaviour using transitions was?	3	4	2	3	2	2
228. To understand how to generate a FSM was?	3	4	3	3	3	3
229. To understand how to describe an adaptive behaviour was?	3	4	3	2	2	3
230. To understand how to describe the activities of an adaptive behaviour as states was?	3	4	3	3	3	2
231. To understand how to describe the sequence of activities for an adaptive behaviour as transitions was?	3	4	3	3	3	2
232. Sabe tool made the process of generating a FSM for the service?	3	4	4	3	3	3
233. Sabe tool made the process of describing an adaptive behaviour?	3	4	3	3	3	3
234. Sabe tool made the process of describing the activities of an adaptive behaviour?	3	4	3	3	3	3
235. Sabe tool made the process of describing the sequence of activities for an adaptive behaviour?	3	4	3	3	3	3
236. When describing adaptive behaviours, would you rate SABE tool as?	3	4	3	3	3	3
321. Aggregating the adaptive behaviours for a composite service was?	3	4	3	3	4	4
322. Viewing the resultant aggregated adaptive behaviours was?	3	4	3	3	3	3
323. To understand how to aggregate the adaptive behaviours for a composite service was?	3	4	3	2	2	3
324. To understand the resultant aggregated adaptive behaviours was?	3	4	3	2	3	2
325. Sabe tool made the process of aggregating the adaptive behaviours for a composite service?	4	4	3	3	3	4
326. Sabe tool made the process of viewing the resultant aggregated adaptive behaviours of a composite service?	4	4	3	3	3	3
327. When aggregating the adaptive behaviours for a composite service, would you rate SABE tool as?	3	4	3	3	3	3
328. Comparing the composite service's FSM with the constituent service's FSM, would you say they are?	4	4	3	3	3	3

Results of the SABE Tool Usability Evaluation Cycle 2

Pre-Test

Users	A	B	C	D	E	F	G	H	I	J	K	L	Z[11]
101. How would you rate your knowledge of Semantic Web Service (Owl-S)?	2	1	2	1	2	1	1	1	2	1	2	1	1

[11] This is the (rounded) average of the pre-test results, where 4 is very positive and 1 is very negative

102. Have you studied about web services?	2	3	2	2	2	2	2	3	2	2	2	1	2
103. Have you used a web service?	1	3	2	2	3	3	3	3	2	2	2	3	3
104. Have you created a web service?	1	2	1	1	1	1	1	3	1	1	1	2	1
105. Have you modelled a web service semantically in Owl-S?	1	1	1	1	1	1	1	1	1	1	1	1	1
106. How would you rate your knowledge of (software) Adaptive Behaviours?	2	2	2	2	2	2	1	2	2	2	1	1	2
107. Have you studied about (software) Adaptive Behaviours?	2	1	2	2	2	1	1	1	2	2	1	1	1
108. Have you used a piece of software with some adaptive behaviour?	2	2	2	1	1	2	1	2	3	4	1	4	2
109. Have you configured an adaptive piece of software or application?	1	2	1	1	1	1	1	1	2	4	1	1	1
110. Have you created a piece of software with some adaptive behaviour?	1	2	1	1	1	1	1	2	1	1	1	1	1
111. How would you rate your knowledge of Finite State Machine?	4	3	2	1	2	2	3	3	3	3	2	4	3
112. Have you studied about Finite State Machine?	4	4	2	1	2	2	3	2	2	3	2	4	2
113. Have you seen a piece of software described as a Finite State Machine?	4	3	1	1	1	1	4	3	3	3	1	4	2
114. Have you described a piece of software semantically?	4	3	1	1	1	1	1	3	1	1	1	4	2
115. Have you described a piece of software using Finite State Machine?	4	3	1	1	1	1	4	2	2	3	1	4	2
116. Have you used an application which uses a tree structure to represent data (such as folders)?	4	4	4	4	4	4	3	4	2	4	4	4	4
117. Have you used an application which uses a table to display data?	3	4	4	4	4	4	3	4	4	4	3	4	4
118. Have you used an application which uses a table to allow user to edit values?	3	4	4	3	4	4	3	4	3	4	4	4	4
301. How would you rate your understand of service composition?	1	2	2	2	2	2	2	2	3	1	2	2	2
302. How would you rate your confidence in using the SABE application to browse a service?	2	2	3	2	1	2	3	2	3	2	2	1	2

Results of the SABE Tool Usability Evaluation Cycle 2

Task-related Questions

Users	A	B	C	D	E	F	G	H	I	J	K	L
121. What is the activity of the ExpensiveModestate1 of the	1	1	1	1	1	1	1	1	1	1	1	1

ExpensiveMode adaptive behaviour?												
122. What is the type of the of the ExpensiveMode adaptive behaviour?	1	1	1	1	1	1	1	1	1	1	1	1
123. What is the guard condition in the transition (between ColourPrintingState1 and ColourPrintingState2) of the ColourPriting adaptive behaviour?	1	1	1	1	0	1	1	1	1	1	0	1
124. What triggers the transition (between ColourPrintingState1 and ColourPrintingState2) of the ColourPriting adaptive behaviour?	1	1	1	1	1	1	1	1	1	1	1	1
321. Does the FSM belonging to the composite service have IdleState, InputState, ProceState, OutputState like of an atomice service?	1	1	1	1	1	1	1	1	1	1	1	1
322. Are all the adaptive behaviours from the constituent services present in the FSM of the composite service?	1	1	1	1	1	1	1	1	1	1	1	1
323. How many adaptive behaviours are present under the ProcessState of the composite service's FSM?	1	1	1	1	1	1	1	1	1	1	1	1

Results of the SABE Tool Usability Evaluation Cycle 2 Questionnaire

Users	A	B	C	D	E	F	G	H	I	J	K	L
124. What triggers the transition (between ColourPrintingState1 and ColourPrintingState2) of the ColourPriting adaptive behaviour?	2	2	2	2	2	2	2	2	2	2	2	2
125. Starting the SABE tool was?	4	3	4	4	3	3	4	4	3	3	3	3
126. Loading the service was?	4	4	4	4	3	3	4	4	3	4	3	3
127. Browsing the service's process (with its inputs, outputs,...) was?	4	4	3	4	3	2	3	3	3	3	3	3
128. Browsing the service's Finite State Machine (FSM) was?	3	3	3	4	3	3	4	3	3	3	3	3
129. Identifying the adaptive behaviours contained in the FSM was?	3	3	3	4	3	3	3	3	3	2	3	3
130. Viewing the details of an adaptive behaviour was?	4	3	3	4	3	3	3	3	3	2	3	3
131. Identifying the actions performed by an adaptive behaviour was?	3	3	3	4	2	3	3	3	3	3	3	3
132. Identifying the sequence of actions performed by an adaptive behaviour was?	3	3	3	3	3	3	3	3	3	2	2	3

133. Sabe tool made the process of browsing the description of an adaptive behaviour?	3	3	3	3	3	3	3	4	3	3	3	3
134. Sabe tool made the process of identifying the actions of an adaptive behaviour?	3	3	3	4	3	3	4	3	3	3	3	3
135. Sabe tool made the process of identifying the sequence of actions of an adaptive behaviour?	3	3	3	3	3	3	3	3	3	2	3	3
136. To understand the description of adaptive behaviours using SABE tool was?	3	3	3	3	2	3	3	3	3	3	2	2
137. To understand the description of states describing an action of adaptive behaviours using SABE tool was?	3	2	3	3	2	3	4	3	3	3	2	2
138. To understand the description of transitions describing a sequence of actions of adaptive behaviours using SABE tool was?	3	2	2	3	2	3	4	3	3	3	2	2
221. Generating a FSM for the service's process was?	3	2	3	4	3	3	3	3	3	3	3	3
222. Creating an adaptive behaviour was?	3	4	3	4	3	3	4	3	3	3	3	3
223. Creating the states for this adaptive behaviour was?	3	4	4	3	3	3	4	3	3	3	3	3
224. Creating the transitions for this adaptive behaviour was?	3	3	4	3	3	3	4	2	3	3	3	3
225. Describing an adaptive behaviour was?	3	4	4	3	3	3	3	3	3	3	2	3
226. Describing the activities of an adaptive behaviour using states was?	2	3	3	3	3	3	4	3	2	3	2	3
227. Describing the sequence of activities for an adaptive behaviour using transitions was?	2	2	3	3	3	3	3	2	2	3	3	3
228. Sabe tool made the process of generating a FSM for the service?	3	3	4	3	3	3	4	3	3	3	3	3
229. Sabe tool made the process of describing an adaptive behaviour?	3	3	3	3	3	3	3	3	2	3	3	3
230. Sabe tool made the process of describing the activities of an adaptive behaviour?	3	3	3	3	3	3	3	3	2	3	3	3
231. Sabe tool made the process of describing the sequence of activities for an adaptive behaviour?	3	3	3	3	3	3	4	3	2	3	3	3
232. To understand the process of generating a FSM was?	3	4	2	3	3	3	4	3	3	3	2	3
233. To understand the process of describing an adaptive behaviour was?	3	4	3	3	3	3	3	3	3	2	2	3
234. To understand the process of describing the activities of an adaptive behaviour as states was?	3	2	3	3	3	3	3	3	3	2	2	3
235. To understand the process of describing the sequence of activities for an adaptive behaviour as transitions was?	3	2	3	3	3	3	3	3	3	2	2	3

324. Aggregating the adaptive behaviours for a composite service was?	4	2	4	3	2	3	4	3	4	3	2	3
325. Viewing the resultant aggregated adaptive behaviours was?	4	4	4	3	2	3	4	3	4	3	2	3
326. Sabe tool made the process of aggregating the adaptive behaviours for a composite service?	4	2	4	4	3	3	4	3	4	3	3	3
327. Sabe tool made the process of browsing the resultant aggregated adaptive behaviours?	4	3	4	4	3	3	4	3	4	3	3	3
328. To understand the process of aggregating the adaptive behaviours for a composite service was?	3	2	3	4	2	3	3	3	3	3	2	2
329. To understand the resultant aggregated adaptive behaviours using the SABE tool was?	3	3	3	4	2	3	4	3	3	3	2	2

Results of the SABE Tool Usability Evaluation Cycle 3

Pre-Test

Users	A	B	C	D	E	F	G	H	I	J	K	L	Z[12]
101. How would you rate your knowledge of Semantic Web Service (Owl-S)?	2	1	2	1	2	1	1	1	2	1	2	1	1
102. Have you studied about web services?	2	3	2	2	2	2	2	3	2	2	2	1	2
103. Have you used a web service?	1	3	2	2	3	3	3	3	2	2	2	3	3
104. Have you created a web service?	1	2	1	1	1	1	1	3	1	1	1	2	1
105. Have you modeled a web service semantically in Owl-S?	1	1	1	1	1	1	1	1	1	1	1	1	1
106. How would you rate your knowledge of (software) Adaptive Behaviours?	2	2	2	2	2	2	1	2	2	2	1	1	2
107. Have you studied about (software) Adaptive Behaviours?	2	1	2	2	2	1	1	1	2	2	1	1	1
108. Have you used a piece of software with some adaptive behaviour?	2	2	2	1	1	2	1	2	3	4	1	4	2
109. Have you configured an adaptive piece of software or application?	1	2	1	1	1	1	1	1	2	4	1	1	1
110. Have you created a piece of software with some adaptive behaviour?	1	2	1	1	1	1	1	2	1	1	1	1	1
111. How would you rate your knowledge of Finite State Machine?	4	3	2	1	2	2	3	3	3	3	2	4	3
112. Have you studied about Finite State Machine?	4	4	2	1	2	2	3	2	2	3	2	4	2

[12] This is the (rounded) average of the pre-test results, where 4 is very positive and 1 is very negative

113. Have you seen a piece of software described as a Finite State Machine?	4	3	1	1	1	1	4	3	3	3	1	4	2
114. Have you described a piece of software semantically?	4	3	1	1	1	1	1	3	1	1	1	4	2
115. Have you described a piece of software using Finite State Machine?	4	3	1	1	1	1	4	2	2	3	1	4	2
116. Have you used an application which uses a tree structure to represent data (such as folders)?	4	4	4	4	4	4	3	4	2	4	4	4	4
117. Have you used an application which uses a table to display data?	3	4	4	4	4	4	3	4	4	4	3	4	4
118. Have you used an application which uses a table to allow user to edit values?	3	4	4	3	4	4	3	4	3	4	4	4	4
301. How would you rate your understand of service composition?	1	2	2	2	2	2	2	2	3	1	2	2	2
302. How would you rate your confidence in using the SABE application to browse a service?	2	2	3	2	1	2	3	2	3	2	2	1	2

Results of the SABE Tool Usability Evaluation Cycle 3

Task-related Questions

Users	A	B	C	D	E	F	G	H	I	J	K	L
121. What is the activity of the ExpensiveModestate1 of the ExpensiveMode adaptive behaviour?	1	1	1	1	1	1	1	1	1	1	1	1
122. What is the type of the of the ExpensiveMode adaptive behaviour?	1	1	1	1	1	1	1	1	1	1	1	1
123. What is the guard condition in the transition (between ColourPrintingState1 and ColourPrintingState2) of the ColourPriting adaptive behaviour?	1	1	1	0	1	0	1	1	1	1	1	1
124. What triggers the transition (between ColourPrintingState1 and ColourPrintingState2) of the ColourPriting adaptive behaviour?	1	1	1	1	1	1	1	1	1	1	1	1
321. Does the FSM belonging to the composite service have IdleState, InputState, ProceState, OutputState like of an atomice service?	1	1	0	1	1	1	1	1	1	1	1	1
322. Are all the adaptive behaviours from the constituent services present in the FSM of the composite service?	1	1	1	1	1	1	1	1	1	1	1	1
323. How many adaptive behaviours are present under the ProcessState of the composite service's FSM?	1	1	1	0	1	1	1	1	1	1	0	0

Results of the SABE Tool Usability Evaluation Cycle 3 Questionnaire

Users	A	B	C	D	E	F	G	H	I	J	K	L
124. What triggers the transition (between ColourPrintingState1 and ColourPrintingState2) of the ColourPriting adaptive behaviour?	2	2	2	2	2	2	2	2	2	2	2	2
125. Starting the SABE tool was?	4	3	4	4	3	4	4	4	4	4	4	3
126. Loading the service was?	4	3	4	4	3	4	4	4	4	4	4	3
127. Browsing the service's process (with its inputs, outputs,...) was?	3	3	4	3	3	4	3	4	3	4	3	2
128. Browsing the service's Finite State Machine (FSM) was?	3	3	4	4	3	4	3	4	3	4	3	2
129. Identifying the adaptive behaviours contained in the FSM was?	3	3	3	3	3	4	2	3	3	4	3	2
130. Viewing the details of an adaptive behaviour was?	3	3	4	3	3	4	3	4	3	3	3	3
131. Identifying the actions performed by an adaptive behaviour was?	3	3	3	4	3	4	3	3	3	3	3	3
132. Identifying the sequence of actions performed by an adaptive behaviour was?	3	3	3	4	3	3	3	3	3	2	3	2
133. Sabe tool made the process of browsing the description of an adaptive behaviour?	3	3	3	4	3	4	3	4	3	4	3	3
134. Sabe tool made the process of identifying the actions of an adaptive behaviour?	3	3	3	3	3	4	2	3	3	2	3	3
135. Sabe tool made the process of identifying the sequence of actions of an adaptive behaviour?	3	3	3	4	3	4	2	3	3	2	3	2
136. To understand the description of adaptive behaviours using SABE tool was?	4	3	3	4	3	4	2	3	3	3	3	3
137. To understand the description of states describing an action of adaptive behaviours using SABE tool was?	4	3	3	3	3	4	2	4	2	2	3	3
138. To understand the description of transitions describing a sequence of actions of adaptive behaviours using SABE tool was?	3	3	3	3	3	3	2	3	2	2	3	2
221. Generating a FSM for the service's process was?	3	3	3	4	3	3	1	3	3	4	3	3
222. Creating an adaptive behaviour was?	3	3	3	3	3	3	1	3	3	4	3	2
223. Creating the states for this adaptive behaviour was?	3	3	3	3	3	3	1	3	3	4	3	2
224. Creating the transitions for this adaptive behaviour was?	3	3	3	3	3	3	1	3	3	3	3	2

225. Describing an adaptive behaviour was?	3	3	3	3	3	3	1	3	3	3	3	3
226. Describing the activities of an adaptive behaviour using states was?	3	3	3	3	3	3	1	3	3	3	3	3
227. Describing the sequence of activities for an adaptive behaviour using transitions was?	3	3	3	3	3	3	1	3	3	3	3	1
228. Sabe tool made the process of generating a FSM for the service?	3	3	3	3	3	3	1	4	3	3	3	2
229. Sabe tool made the process of describing an adaptive behaviour?	3	3	3	3	3	3	1	4	3	3	3	3
230. Sabe tool made the process of describing the activities of an adaptive behaviour?	3	3	3	3	3	3	1	4	3	3	3	3
231. Sabe tool made the process of describing the sequence of activities for an adaptive behaviour?	3	3	3	3	3	3	1	4	3	3	3	2
232. To understand the process of generating a FSM was?	4	3	3	3	3	3	1	4	3	3	3	2
233. To understand the process of describing an adaptive behaviour was?	4	3	3	3	3	3	1	4	3	3	3	2
234. To understand the process of describing the activities of an adaptive behaviour as states was?	4	3	3	3	3	3	1	4	2	3	3	2
235. To understand the process of describing the sequence of activities for an adaptive behaviour as transitions was?	4	3	3	3	3	3	1	4	2	3	3	2
324. Aggregating the adaptive behaviours for a composite service was?	4	3	3	3	3	4	1	4	4	3	4	2
325. Viewing the resultant aggregated adaptive behaviours was?	4	3	3	3	3	4	1	4	4	3	3	2
326. Sabe tool made the process of aggregating the adaptive behaviours for a composite service?	4	3	3	3	3	4	1	4	4	3	3	3
327. Sabe tool made the process of browsing the resultant aggregated adaptive behaviours?	4	3	3	3	3	4	1	4	4	3	3	2
328. To understand the process of aggregating the adaptive behaviours for a composite service was?	3	3	3	2	3	4	1	4	3	3	3	2
329. To understand the resultant aggregated adaptive behaviours using the SABE tool was?	3	3	3	3	3	4	1	4	3	3	3	1

Analysis of the SABE Tool Usability Evaluations

SABE Usability Evaluation Cycle 1

Usability Test 1	Easiness		Helpfulness		Comprehension	
Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*

Very Difficult	1	1	1	0	1	0
Difficult	2	6	2	2	2	4
Easy	3	18	3	13	3	11
Very Easy	4	5	4	3	4	3

Usability Test 2	**Easiness**		**Helpfulness**		**Comprehension**	
Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*
Very Difficult	1	0	1	0	1	0
Difficult	2	4	2	1	2	6
Easy	3	23	3	23	3	14
Very Easy	4	9	4	6	4	4

Usability Test 3	**Easiness**		**Helpfulness**		**Comprehension**	
Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*
Very Difficult	1	0	1	0	1	0
Difficult	2	0	2	0	2	4
Easy	3	8	3	7	3	6
Very Easy	4	4	4	5	4	2

SABE Usability Evaluation Cycle 2

Usability Test 1	**Easiness**		**Helpfulness**		**Comprehension**	
Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*
Very Difficult	1	0	1	0	1	0
Difficult	2	5	2	1	2	12
Easy	3	49	3	32	3	22

Very Easy	4	6	4	3	4	2

Usability Test 2	Easiness		Helpfulness		Comprehension	
Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*
Very Difficult	1	0	1	0	1	0
Difficult	2	10	2	3	2	10
Easy	3	62	3	42	3	35
Very Easy	4	12	4	3	4	3

Usability Test 3	Easiness		Helpfulness		Comprehension	
Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*
Very Difficult	1	0	1	0	1	0
Difficult	2	5	2	1	2	7
Easy	3	10	3	13	3	14
Very Easy	4	9	4	10	4	3

SABE Usability Evaluation Cycle 3

Usability Test 1	Easiness		Helpfulness		Comprehension	
Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*
Very Difficult	1	0	1	0	1	0
Difficult	2	5	2	5	2	8
Easy	3	42	3	24	3	22
Very Easy	4	13	4	7	4	6

Usability Test 2	Easiness		Helpfulness		Comprehension	

Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*
Very Difficult	1	8	1	4	1	4
Difficult	2	3	2	2	2	6
Easy	3	69	3	38	3	30
Very Easy	4	4	4	4	4	8

Usability Test 3	**Easiness**		**Helpfulness**		**Comprehension**	
Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*
Very Difficult	1	2	1	2	1	3
Difficult	2	2	2	1	2	2
Easy	3	11	3	13	3	15
Very Easy	4	9	4	8	4	4

SABE Comparison of Usability Evaluation Cycles

Usability Test 1	**Easiness**			
Observed	**A**	**B**	**C**	**Total**
Easy	23	55	55	133
Difficult	7	5	5	17
	30	60	60	150
Expected	**A**	**B**	**C**	
Easy	26.6	53.2	53.2	
Difficult	3.4	6.8	6.8	
Null Hypothesis	**Observed**	**Expected**	**Result**	
P-value	0.068094	0.05	***TRUE***	

Usability Test 1	**Helpfulness**			
Observed	**A**	**B**	**C**	**Total**
Easy	16	35	31	82
Difficult	2	1	5	8
	18	36	36	90
Expected	**A**	**B**	**C**	
Easy	16.4	32.8	32.8	
Difficult	1.6	3.2	3.2	
Null Hypothesis	**Observed**	**Expected**	**Result**	
P-value	0.236798	0.05	***TRUE***	

Usability Test 1	**Comprehension**			
Observed	**A**	**B**	**C**	**Total**
Easy	14	24	28	66
Difficult	4	12	8	24
	18	36	36	90
Expected	**A**	**B**	**C**	
Easy	13.2	26.4	26.4	
Difficult	4.8	9.6	9.6	
Null Hypothesis	**Observed**	**Expected**	**Result**	
P-value	0.505697	0.05	***TRUE***	

Usability Test 2	**Easiness**			
Observed	**A**	**B**	**C**	**Total**
Easy	32	74	*73*	179

Difficult	4	10	*11*	25
	36	84	*84*	204
Expected	**A**	**B**	**C**	
Easy	31.58824	73.70588	*73.70588*	
Difficult	4.411765	10.29412	*10.29412*	
Null Hypothesis	**Observed**	**Expected**	**Result**	
P-value	0.947177	0.05	***TRUE***	

Usability Test 2	**Helpfulness**			
Observed	**A**	**B**	**C**	**Total**
Easy	29	45	*42*	116
Difficult	1	3	*6*	10
	30	48	*48*	126
Expected	**A**	**B**	**C**	
Easy	27.61905	44.19048	*44.19047619*	
Difficult	2.380952	3.809524	*3.80952381*	
Null Hypothesis	**Observed**	**Expected**	**Result**	
P-value	0.297459	0.05	***TRUE***	

Usability Test 2	**Comprehension**			
Observed	**A**	**B**	**C**	**Total**
Easy	18	38	*38*	94
Difficult	6	10	*10*	26
	24	48	*48*	120
Expected	**A**	**B**	**C**	

Easy	18.8	37.6	*37.6*	
Difficult	5.2	10.4	*10.4*	
Null Hypothesis	**Observed**	**Expected**	**Result**	
P-value	0.906468	0.05	***TRUE***	

Usability Test 3	**Easiness**			
Observed	**A**	**B**	**C**	**Total**
Easy	12	19	*20*	51
Difficult	0	5	*4*	9
	12	24	*24*	60
Expected	**A**	**B**	**C**	
Easy	10.2	20.4	*20.4*	
Difficult	1.8	3.6	*3.6*	
Null Hypothesis	**Observed**	**Expected**	**Result**	
P-value	0.245311	0.05	***TRUE***	

Usability Test 3	**Helpfulness**			
Observed	**A**	**B**	**C**	**Total**
Easy	12	23	*21*	56
Difficult	0	1	*3*	4
	12	24	*24*	60
Expected	**A**	**B**	**C**	
Easy	11.2	22.4	*22.4*	
Difficult	0.8	1.6	*1.6*	
Null Hypothesis	**Observed**	**Expected**	**Result**	

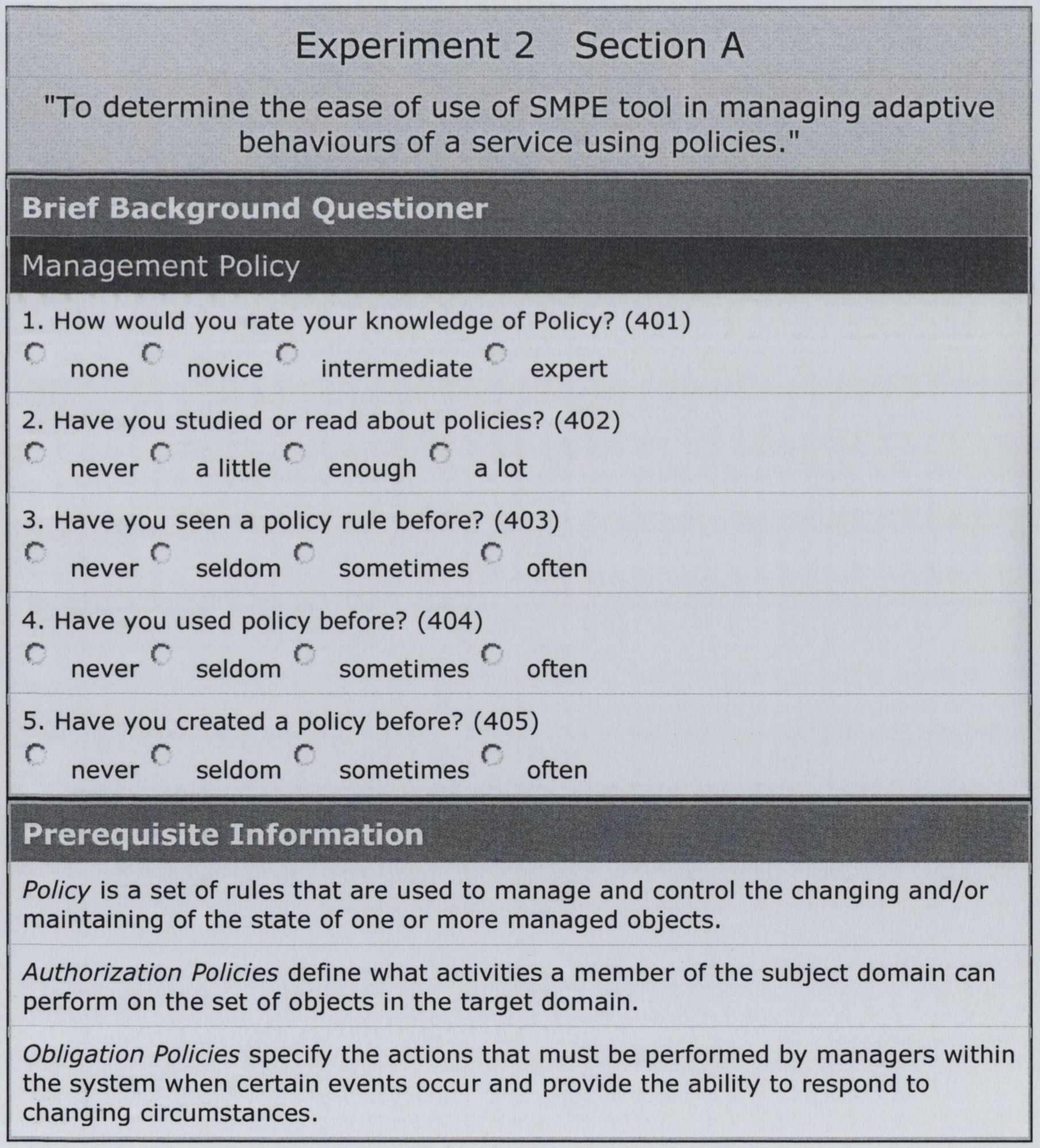
P-value	0.299585	0.05	***TRUE***	

Usability Test 3	**Comprehension**			
Observed	**A**	**B**	**C**	**Total**
Easy	8	17	*19*	44
Difficult	4	7	*5*	16
	12	24	*24*	60
Expected	**A**	**B**	**C**	
Easy	8.8	17.6	*17.6*	
Difficult	3.2	6.4	*6.4*	
Null Hypothesis	**Observed**	**Expected**	**Result**	
P-value	0.681457	0.05	***TRUE***	

Appendix C – SMPE Tool Usability Evaluation Experiment

The SMPE tool usability evaluation presented to the users, which consists of three usability tests. Each usability test presents the instructions, experiment tasks, and the usability questionnaire.

Instructions and Questionnaire for SMPE Usability Test 1



Experiment 2 Section A

"To determine the ease of use of SMPE tool in managing adaptive behaviours of a service using policies."

Brief Background Questioner

Management Policy

1. How would you rate your knowledge of Policy? (401)
○ none ○ novice ○ intermediate ○ expert

2. Have you studied or read about policies? (402)
○ never ○ a little ○ enough ○ a lot

3. Have you seen a policy rule before? (403)
○ never ○ seldom ○ sometimes ○ often

4. Have you used policy before? (404)
○ never ○ seldom ○ sometimes ○ often

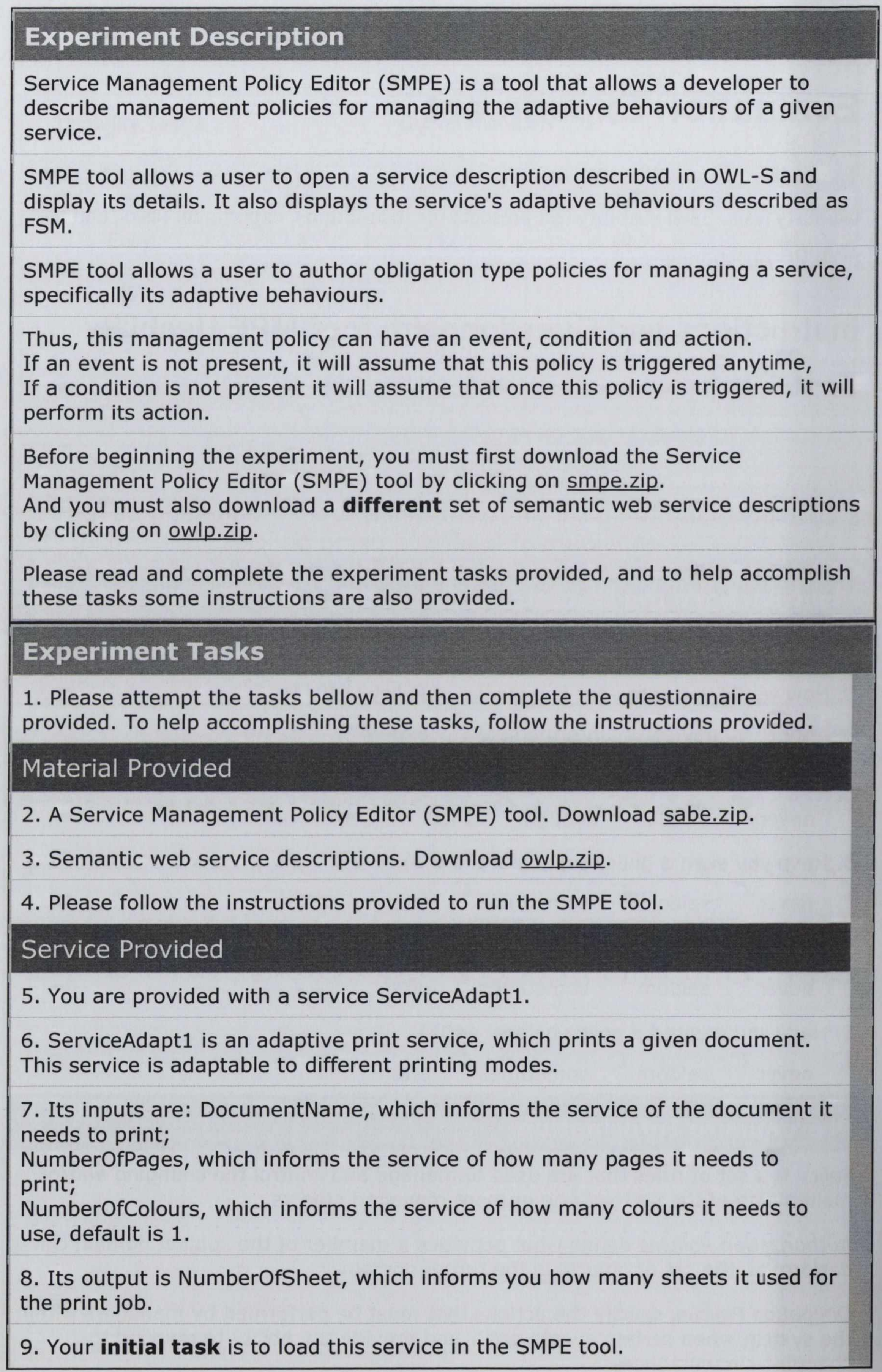
5. Have you created a policy before? (405)
○ never ○ seldom ○ sometimes ○ often

Prerequisite Information

Policy is a set of rules that are used to manage and control the changing and/or maintaining of the state of one or more managed objects.

Authorization Policies define what activities a member of the subject domain can perform on the set of objects in the target domain.

Obligation Policies specify the actions that must be performed by managers within the system when certain events occur and provide the ability to respond to changing circumstances.



Experiment Description

Service Management Policy Editor (SMPE) is a tool that allows a developer to describe management policies for managing the adaptive behaviours of a given service.

SMPE tool allows a user to open a service description described in OWL-S and display its details. It also displays the service's adaptive behaviours described as FSM.

SMPE tool allows a user to author obligation type policies for managing a service, specifically its adaptive behaviours.

Thus, this management policy can have an event, condition and action.
If an event is not present, it will assume that this policy is triggered anytime,
If a condition is not present it will assume that once this policy is triggered, it will perform its action.

Before beginning the experiment, you must first download the Service Management Policy Editor (SMPE) tool by clicking on smpe.zip.
And you must also download a **different** set of semantic web service descriptions by clicking on owlp.zip.

Please read and complete the experiment tasks provided, and to help accomplish these tasks some instructions are also provided.

Experiment Tasks

1. Please attempt the tasks bellow and then complete the questionnaire provided. To help accomplishing these tasks, follow the instructions provided.

Material Provided

2. A Service Management Policy Editor (SMPE) tool. Download sabe.zip.

3. Semantic web service descriptions. Download owlp.zip.

4. Please follow the instructions provided to run the SMPE tool.

Service Provided

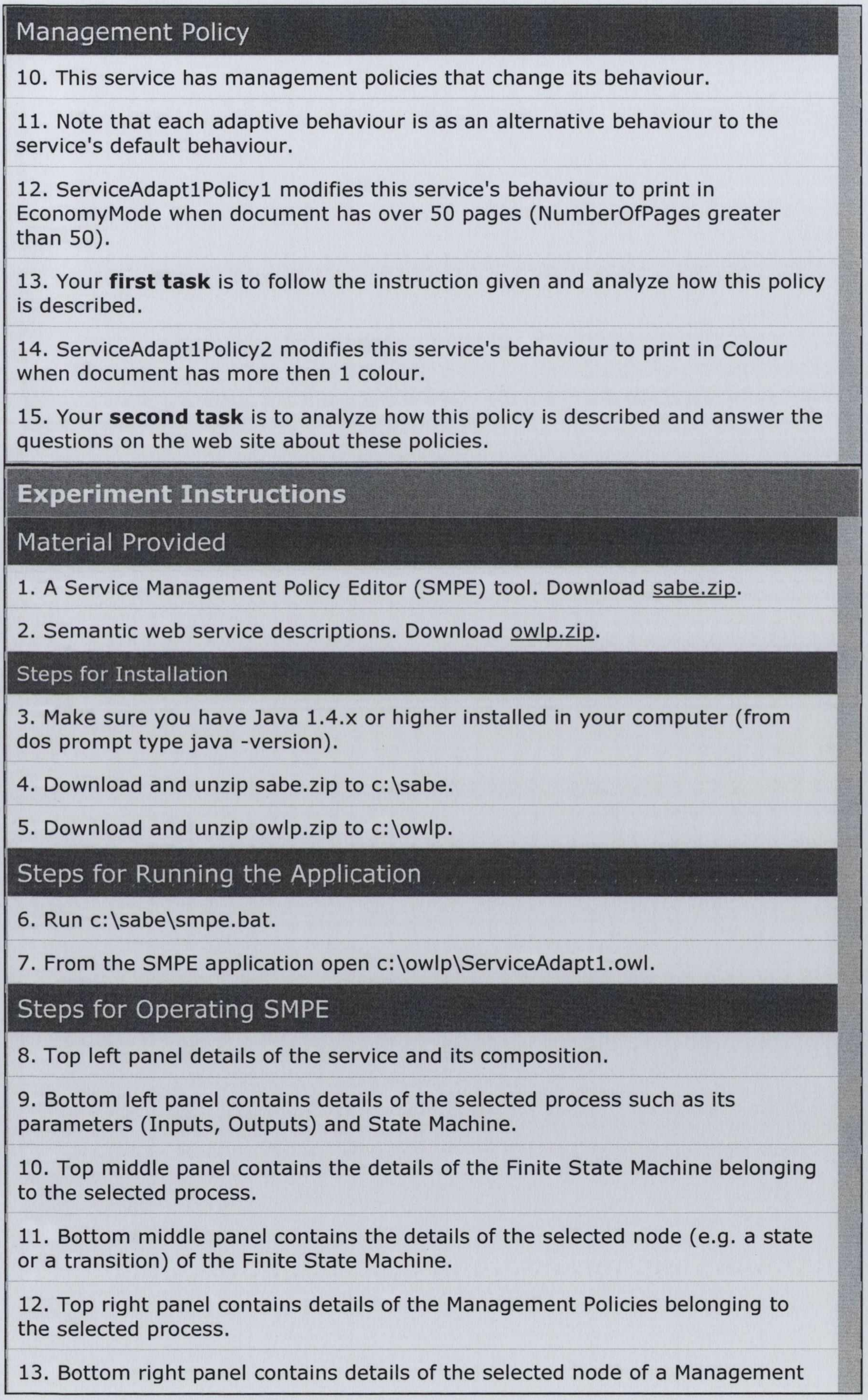
5. You are provided with a service ServiceAdapt1.

6. ServiceAdapt1 is an adaptive print service, which prints a given document. This service is adaptable to different printing modes.

7. Its inputs are: DocumentName, which informs the service of the document it needs to print;
NumberOfPages, which informs the service of how many pages it needs to print;
NumberOfColours, which informs the service of how many colours it needs to use, default is 1.

8. Its output is NumberOfSheet, which informs you how many sheets it used for the print job.

9. Your **initial task** is to load this service in the SMPE tool.



Management Policy

10. This service has management policies that change its behaviour.

11. Note that each adaptive behaviour is as an alternative behaviour to the service's default behaviour.

12. ServiceAdapt1Policy1 modifies this service's behaviour to print in EconomyMode when document has over 50 pages (NumberOfPages greater than 50).

13. Your **first task** is to follow the instruction given and analyze how this policy is described.

14. ServiceAdapt1Policy2 modifies this service's behaviour to print in Colour when document has more then 1 colour.

15. Your **second task** is to analyze how this policy is described and answer the questions on the web site about these policies.

Experiment Instructions

Material Provided

1. A Service Management Policy Editor (SMPE) tool. Download sabe.zip.

2. Semantic web service descriptions. Download owlp.zip.

Steps for Installation

3. Make sure you have Java 1.4.x or higher installed in your computer (from dos prompt type java -version).

4. Download and unzip sabe.zip to c:\sabe.

5. Download and unzip owlp.zip to c:\owlp.

Steps for Running the Application

6. Run c:\sabe\smpe.bat.

7. From the SMPE application open c:\owlp\ServiceAdapt1.owl.

Steps for Operating SMPE

8. Top left panel details of the service and its composition.

9. Bottom left panel contains details of the selected process such as its parameters (Inputs, Outputs) and State Machine.

10. Top middle panel contains the details of the Finite State Machine belonging to the selected process.

11. Bottom middle panel contains the details of the selected node (e.g. a state or a transition) of the Finite State Machine.

12. Top right panel contains details of the Management Policies belonging to the selected process.

13. Bottom right panel contains details of the selected node of a Management

Policy.

14. Bottom panel contains the details of the services dataflow.

15. Double click on a node or click the plus sign to expand a node of a tree.

16. Hover over for information on the node via a tool tip.

17. Right click on a node to perform action if any.

Steps for Selecting Service

18. Expand the service node on the left panel; a process will appear under it.

19. Select the process, its description will appear in the panel below and its policy on the top right panel.

Steps for Browsing Management Policies

20. Expand Policy node on the top right panel and select the policy node (ProcessAdapt1Policy1).

21. This policy's details will be shown in the bottom right panel, where you can see its name, type and description.

22. Expand the policy node (ProcessAdapt1Policy1) and you will see Events node, Conditions node, and Actions node.

23. Expand Events node and select the event node (ProcessAdapt1Policy1Event1).

24. In the bottom right panel it is possible to see the name, type and value of the selected event.

25. Expand Conditions node and select the action node (ProcessAdapt1Policy1Condition1).

26. In the bottom right panel it is possible to see the name, type, subject, predicate and value of the selected condition.

27. Policy's condition should be NumberOfPages greater than 50.

28. Expand Actions node and select the action node (ProcessAdapt1Policy1Action1).

29. In the bottom right panel it is possible to see the name, type and value of the selected action.

30. Policy's action should be EconomyMode.

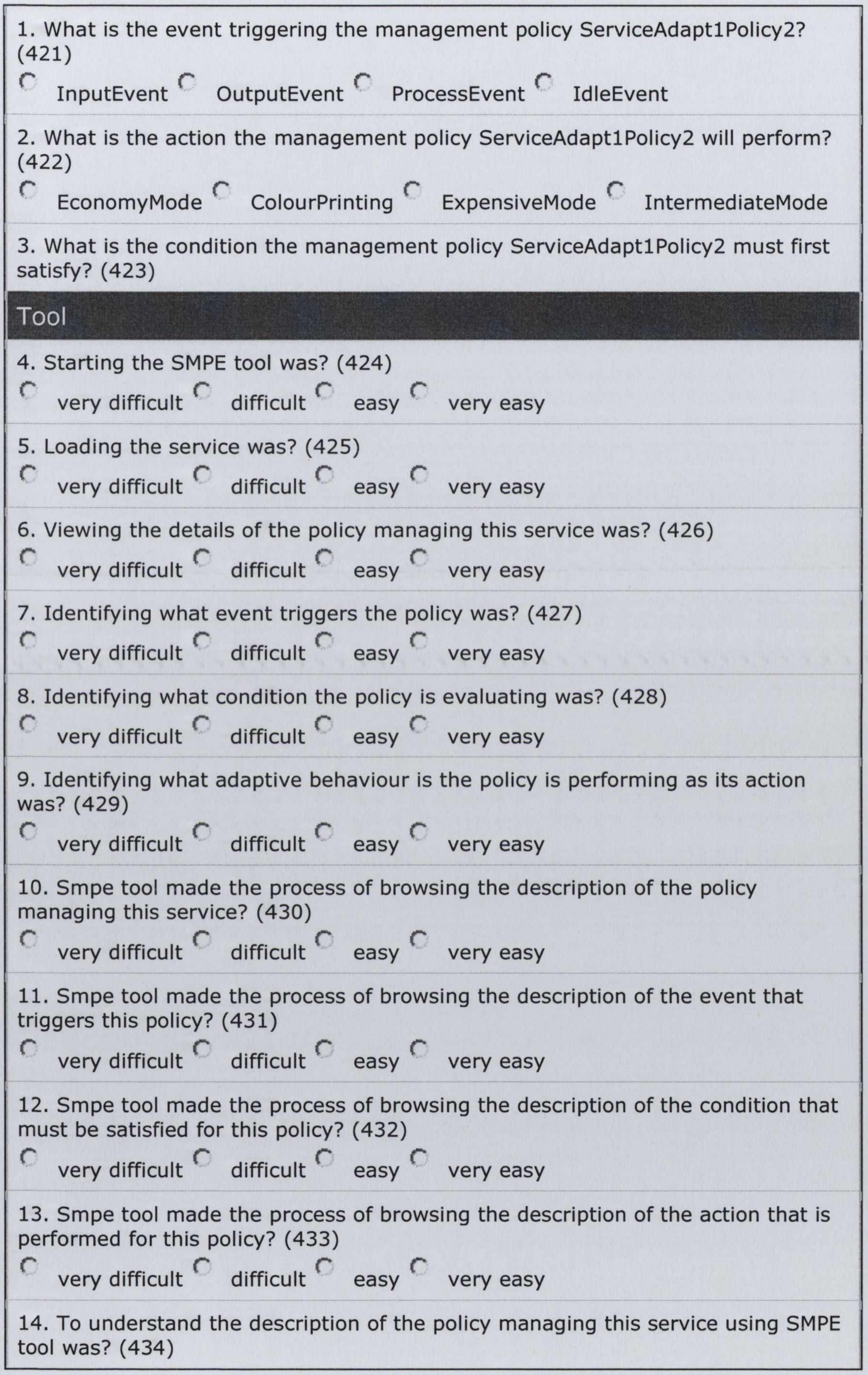
31. Now it is your turn to analyse the second policy (ProcessAdapt1Policy2).

Steps for Closing your Service

32. Close your service by clicking on the X in the inner frame and answering yes when prompted.

Experiment Questionnaire

Task


1. What is the event triggering the management policy ServiceAdapt1Policy2? (421)

○ InputEvent ○ OutputEvent ○ ProcessEvent ○ IdleEvent

2. What is the action the management policy ServiceAdapt1Policy2 will perform? (422)

○ EconomyMode ○ ColourPrinting ○ ExpensiveMode ○ IntermediateMode

3. What is the condition the management policy ServiceAdapt1Policy2 must first satisfy? (423)

Tool

4. Starting the SMPE tool was? (424)

○ very difficult ○ difficult ○ easy ○ very easy

5. Loading the service was? (425)

○ very difficult ○ difficult ○ easy ○ very easy

6. Viewing the details of the policy managing this service was? (426)

○ very difficult ○ difficult ○ easy ○ very easy

7. Identifying what event triggers the policy was? (427)

○ very difficult ○ difficult ○ easy ○ very easy

8. Identifying what condition the policy is evaluating was? (428)

○ very difficult ○ difficult ○ easy ○ very easy

9. Identifying what adaptive behaviour is the policy is performing as its action was? (429)

○ very difficult ○ difficult ○ easy ○ very easy

10. Smpe tool made the process of browsing the description of the policy managing this service? (430)

○ very difficult ○ difficult ○ easy ○ very easy

11. Smpe tool made the process of browsing the description of the event that triggers this policy? (431)

○ very difficult ○ difficult ○ easy ○ very easy

12. Smpe tool made the process of browsing the description of the condition that must be satisfied for this policy? (432)

○ very difficult ○ difficult ○ easy ○ very easy

13. Smpe tool made the process of browsing the description of the action that is performed for this policy? (433)

○ very difficult ○ difficult ○ easy ○ very easy

14. To understand the description of the policy managing this service using SMPE tool was? (434)

◌ very confusing ◌ confusing ◌ clear ◌ very clear

15. To understand the description of the event that triggers this policy using SMPE tool was? (435)

◌ very confusing ◌ confusing ◌ clear ◌ very clear

16. To understand the description of the condition that must be satisfied for this policy using SMPE tool was? (436)

◌ very confusing ◌ confusing ◌ clear ◌ very clear

17. To understand the description of the action that is performed for this policy using SMPE tool was? (437)

◌ very confusing ◌ confusing ◌ clear ◌ very clear

18. Comment on any feature of SABE tool you found useful? (438)

19. Comment on any feature of SABE tool you found difficult? (439)

Instructions and Questionnaire for SMPE Usability Test 2

Experiment 2 Section B

"To determine the ease of use of SMPE tool in defining a policy to manage the adaptive behaviours of a composite service."

Brief Background Questioner

Management Policy

1. How would you rate your confidence in describing a management policy? (501)

◌ unsure ◌ some confident ◌ confident ◌ very confident

Prerequisite Information

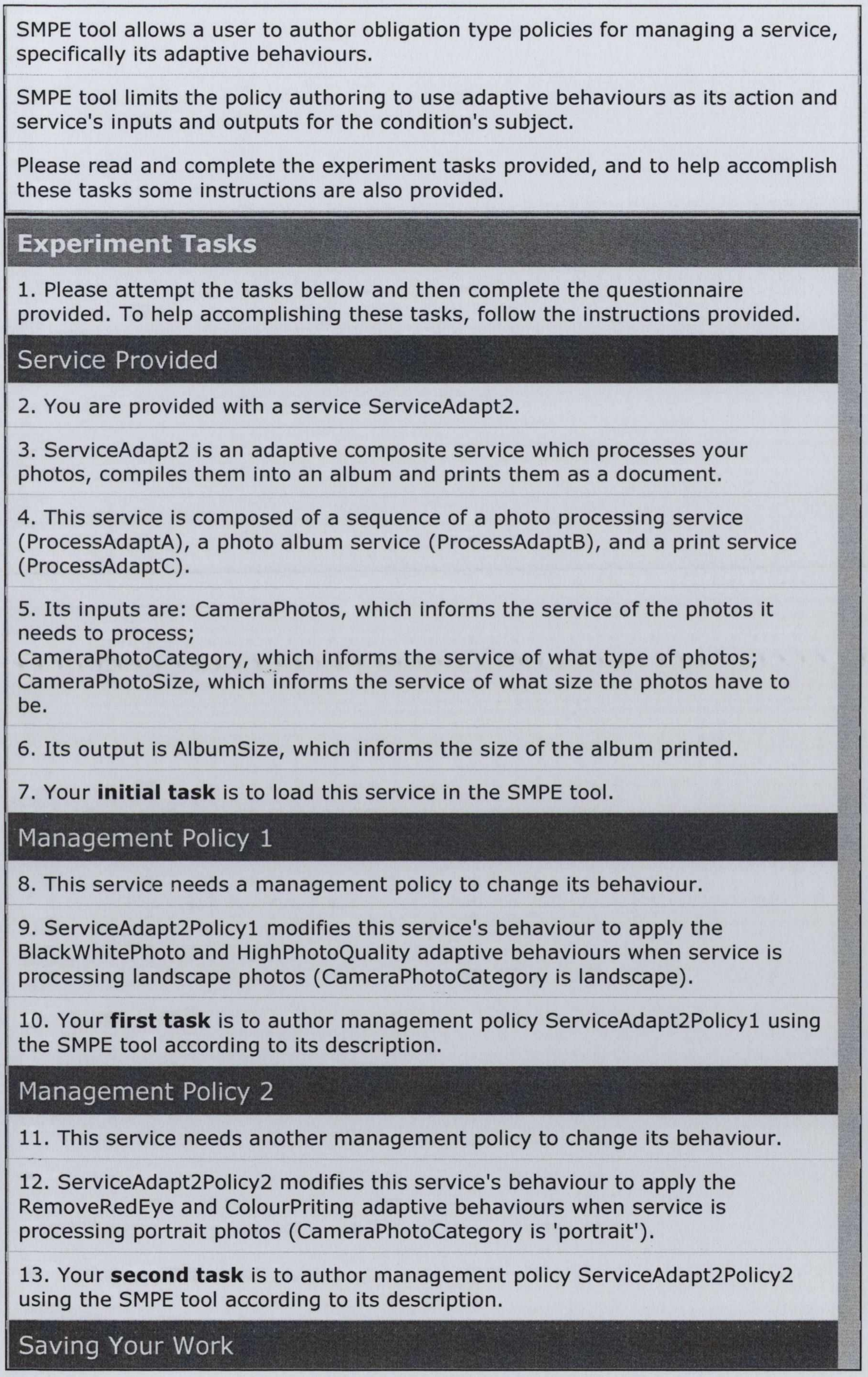
Policy Events are the events in which will trigger the policy.

Policy Conditions are the conditions that must be satisfied before the policy perform its actions.

Policy Actions are the actions that the policy performs once its conditions are satisfied.

Experiment Description

Service Management Policy Editor (SMPE) is a tool that allows a developer to describe management policies for managing the adaptive behaviours of a given service.



SMPE tool allows a user to author obligation type policies for managing a service, specifically its adaptive behaviours.

SMPE tool limits the policy authoring to use adaptive behaviours as its action and service's inputs and outputs for the condition's subject.

Please read and complete the experiment tasks provided, and to help accomplish these tasks some instructions are also provided.

Experiment Tasks

1. Please attempt the tasks bellow and then complete the questionnaire provided. To help accomplishing these tasks, follow the instructions provided.

Service Provided

2. You are provided with a service ServiceAdapt2.

3. ServiceAdapt2 is an adaptive composite service which processes your photos, compiles them into an album and prints them as a document.

4. This service is composed of a sequence of a photo processing service (ProcessAdaptA), a photo album service (ProcessAdaptB), and a print service (ProcessAdaptC).

5. Its inputs are: CameraPhotos, which informs the service of the photos it needs to process;
CameraPhotoCategory, which informs the service of what type of photos;
CameraPhotoSize, which informs the service of what size the photos have to be.

6. Its output is AlbumSize, which informs the size of the album printed.

7. Your **initial task** is to load this service in the SMPE tool.

Management Policy 1

8. This service needs a management policy to change its behaviour.

9. ServiceAdapt2Policy1 modifies this service's behaviour to apply the BlackWhitePhoto and HighPhotoQuality adaptive behaviours when service is processing landscape photos (CameraPhotoCategory is landscape).

10. Your **first task** is to author management policy ServiceAdapt2Policy1 using the SMPE tool according to its description.

Management Policy 2

11. This service needs another management policy to change its behaviour.

12. ServiceAdapt2Policy2 modifies this service's behaviour to apply the RemoveRedEye and ColourPriting adaptive behaviours when service is processing portrait photos (CameraPhotoCategory is 'portrait').

13. Your **second task** is to author management policy ServiceAdapt2Policy2 using the SMPE tool according to its description.

Saving Your Work

14. Now that you have finished authoring these management policies, your **final task** is to save them by clicking on File and then save.

Experiment Instructions

Material Provided

1. A Service Management Policy Editor (SMPE) tool. Download sabe.zip.

2. Semantic web service descriptions. Download owlp.zip.

Steps for Running the Application

3. Run c:\sabe\smpe.bat.

4. From the SMPE application open c:\owlp\ServiceAdapt2.owl.

Steps for Selecting Service

5. Expand the service node on the left panel; a process will appear under it.

6. Select the process, its description will appear in the panel bellow and its FSM on the middle panel.

Steps for Adding Management Policy 1

7. Right click Policy node in the top right panel and select add policy.

8. Expand the newly added policy node.

Steps for Adding Policy Event

9. Right click on Events and select add event to add an event.

10. Expand Events and select the newly added event.

11. Rename this event if you so desire.

12. Select the appropriate value (ProcessEvent) from a list of events for this policy's event.

Steps for Adding Policy Condition

13. Right click on Conditions and select add condition to add an condition.

14. Expand Conditions and select the newly added condition.

15. Rename this condition if you so desire.

16. Select the appropriate subject (CameraPhotoCategory) from a list of subjects for this policy's condition.

17. Select the appropriate predicate (equal) from a list of predicates for this policy's condition.

18. Set the appropriate value (landscape) for this policy's condition.

Steps for Adding Policy Action

19. Right click on Actions and select add action to add an action.

20. Expand Actions and select the newly added action.

21. Rename this action if you so desire.

22. In this case we need to set this action type to complex action.

23. Right click on the action node and add a Boolean.

24. Make sure the Boolean node is an AND Boolean node.

25. Expand the action node and right click the Boolean node to add 2 action nodes.

26. For the first action, select the appropriate value (BlackWhitePhoto) from a list of adaptive behaviours.

27. For the second action, select the appropriate value (HighPhotoQuality) from a list of adaptive behaviours.

Steps for Adding Management Policy 2

28. Repeat the same procedure for adding the management policy ServiceAdapt2Policy2.

29. Use the appropriate names and values (mentioned in the task section).

Steps for Saving your Management Policies

30. Click on File and then save. Make sure there are no errors and click OK.

Steps for Closing your Service

31. Close your service by clicking on the X in the inner frame and answering yes when prompted.

Experiment Questionnaire

Tool

1. Creating a policy for managing the service was? (521)
○ very difficult ○ difficult ○ easy ○ very easy

2. Creating an event that would trigger this policy was? (522)
○ very difficult ○ difficult ○ easy ○ very easy

3. Creating a condition that must be satisfied for this policy was? (523)
○ very difficult ○ difficult ○ easy ○ very easy

4. Creating an action that would be performed by this policy was? (524)
○ very difficult ○ difficult ○ easy ○ very easy

5. Smpe tool made the process of creating the policy for managing the service? (525)
○ very difficult ○ difficult ○ easy ○ very easy

6. Smpe tool made the process of describing the event that would trigger this

policy? (526)

○ very difficult ○ difficult ○ easy ○ very easy

7. Smpe tool made the process of describing the condition that must be satisfied for this policy? (527)

○ very difficult ○ difficult ○ easy ○ very easy

8. Smpe tool made the process of describing the action that would be performed by this policy? (528)

○ very difficult ○ difficult ○ easy ○ very easy

9. To understand the process of creating a policy for managing the service using SMPE tool was? (529)

○ very confusing ○ confusing ○ clear ○ very clear

10. To understand the process of describing the event that would trigger this policy using SMPE tool was? (530)

○ very confusing ○ confusing ○ clear ○ very clear

11. To understand the process of describing the condition that must be satisfied for this policy using SMPE tool was? (531)

○ very confusing ○ confusing ○ clear ○ very clear

12. To understand the process of describing the action that would be performed by this policy using SMPE tool was? (532)

○ very confusing ○ confusing ○ clear ○ very clear

13. Comment on any feature of SABE tool you found useful? (533)

14. Comment on any feature of SABE tool you found difficult? (534)

Instructions and Questionnaire for SMPE Usability Test 3

Experiment 2 Section C

"To determine the ease of use and ease of comprehension of the SMPE tool in refining policies to manage the constituent services of a composite service."

Brief Background Questioner

Management Policy

1. How would you rate your confidence in describing a management policy? (601)

○ unsure ○ some confident ○ confident ○ very confident

Prerequisite Information

Policy Refinement is the taken in as input a policy and generating policy(s) which affect the implementation of that service and any constituent service of that service thus policy refinement is capable of mapping high level goals of a service down to automatically generating low level policies which interact with previously existing management functions in its constituent services.

Experiment Description

Service Management Policy Editor (SMPE) is a tool that allows a developer to describe management policies for managing the adaptive behaviours of a given service.

SMPE tool allows a service provider to refine authored management policies for a given service into discrete policies that will modify the behaviour of this service.

Policy refinement is done according to the description of the adaptive behaviour used by the management policy.

Please read and complete the experiment tasks provided, and to help accomplish these tasks some instructions are also provided.

Experiment Tasks

1. Please attempt the tasks bellow and then complete the questionnaire provided. To help accomplishing these tasks, follow the instructions provided.

Service Provided

2. You are provided with a service ServiceAdapt3.

3. ServiceAdapt3 is an adaptive composite service which processes your photos, compiles them into an album and prints them as a document.

4. This service is composed of a sequence of a photo processing service (ProcessAdaptA), a photo album service (ProcessAdaptB), and a print service (ProcessAdaptC).

5. Its inputs are: CameraPhotos, which informs the service of the photos it needs to process;
CameraPhotoCategory, which informs the service of what type of photos;
CameraPhotoSize, which informs the service of what size the photos have to be.

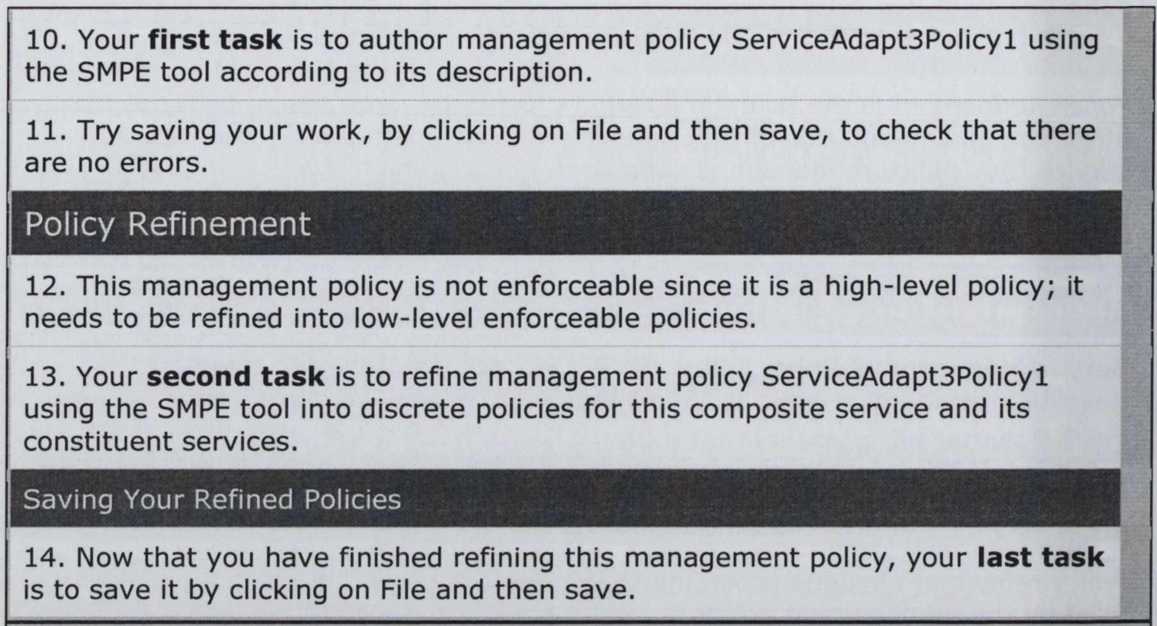
6. Its output is AlbumSize, which informs the size of the album printed.

7. Your **initial task** is to load this service in the SMPE tool.

Management Policy

8. This service has a management policy that changes its behaviour.

9. ServiceAdapt3Policy1 is a policy to create a calendar album of portrait photos by modifying this service's behaviour to apply the RemoveRedEye and CreateCalendar adaptive behaviours when processing portrait photos (CameraPhotoCategory is 'portrait').



10. Your **first task** is to author management policy ServiceAdapt3Policy1 using the SMPE tool according to its description.

11. Try saving your work, by clicking on File and then save, to check that there are no errors.

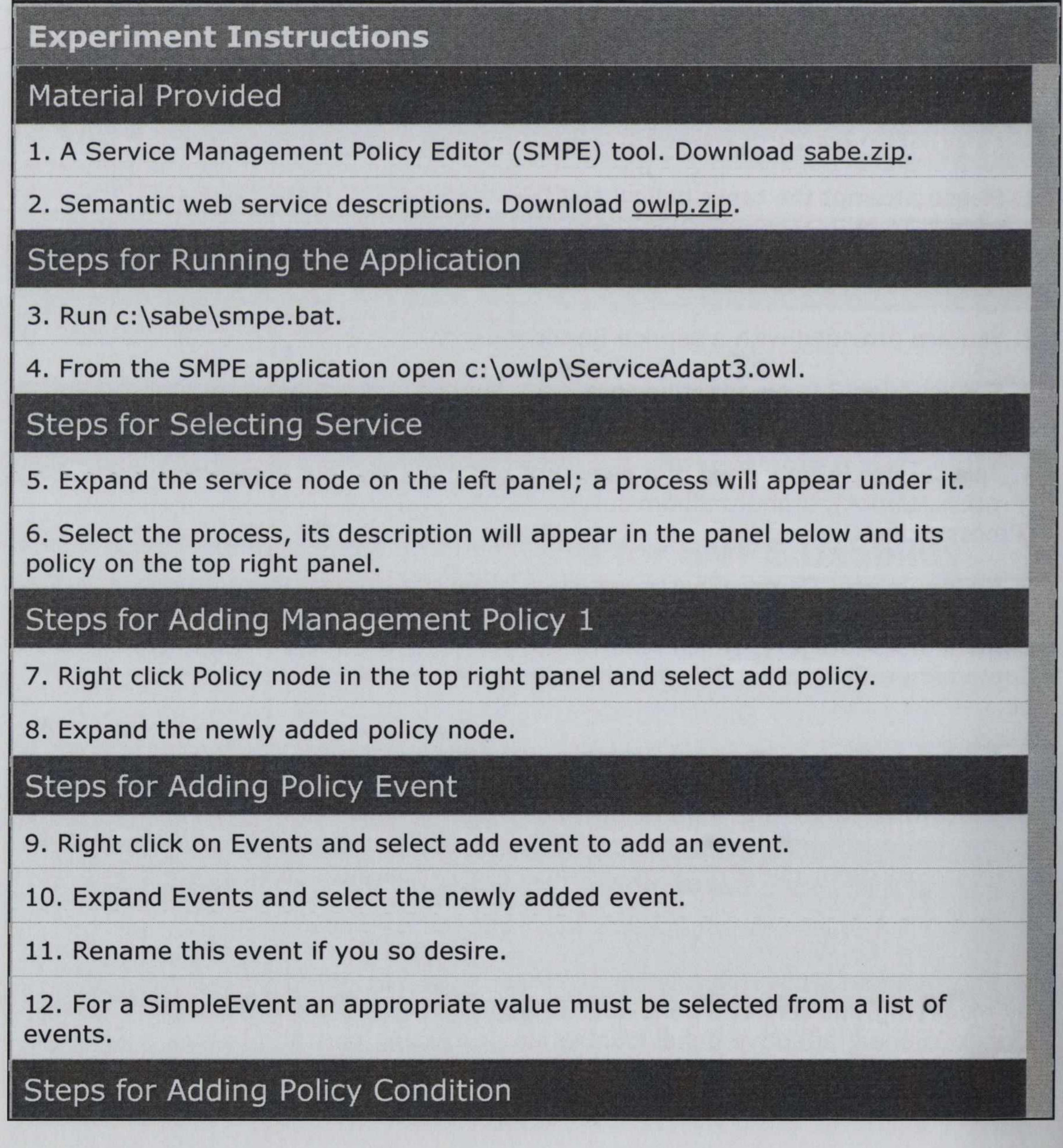
Policy Refinement

12. This management policy is not enforceable since it is a high-level policy; it needs to be refined into low-level enforceable policies.

13. Your **second task** is to refine management policy ServiceAdapt3Policy1 using the SMPE tool into discrete policies for this composite service and its constituent services.

Saving Your Refined Policies

14. Now that you have finished refining this management policy, your **last task** is to save it by clicking on File and then save.



Experiment Instructions

Material Provided

1. A Service Management Policy Editor (SMPE) tool. Download sabe.zip.

2. Semantic web service descriptions. Download owlp.zip.

Steps for Running the Application

3. Run c:\sabe\smpe.bat.

4. From the SMPE application open c:\owlp\ServiceAdapt3.owl.

Steps for Selecting Service

5. Expand the service node on the left panel; a process will appear under it.

6. Select the process, its description will appear in the panel below and its policy on the top right panel.

Steps for Adding Management Policy 1

7. Right click Policy node in the top right panel and select add policy.

8. Expand the newly added policy node.

Steps for Adding Policy Event

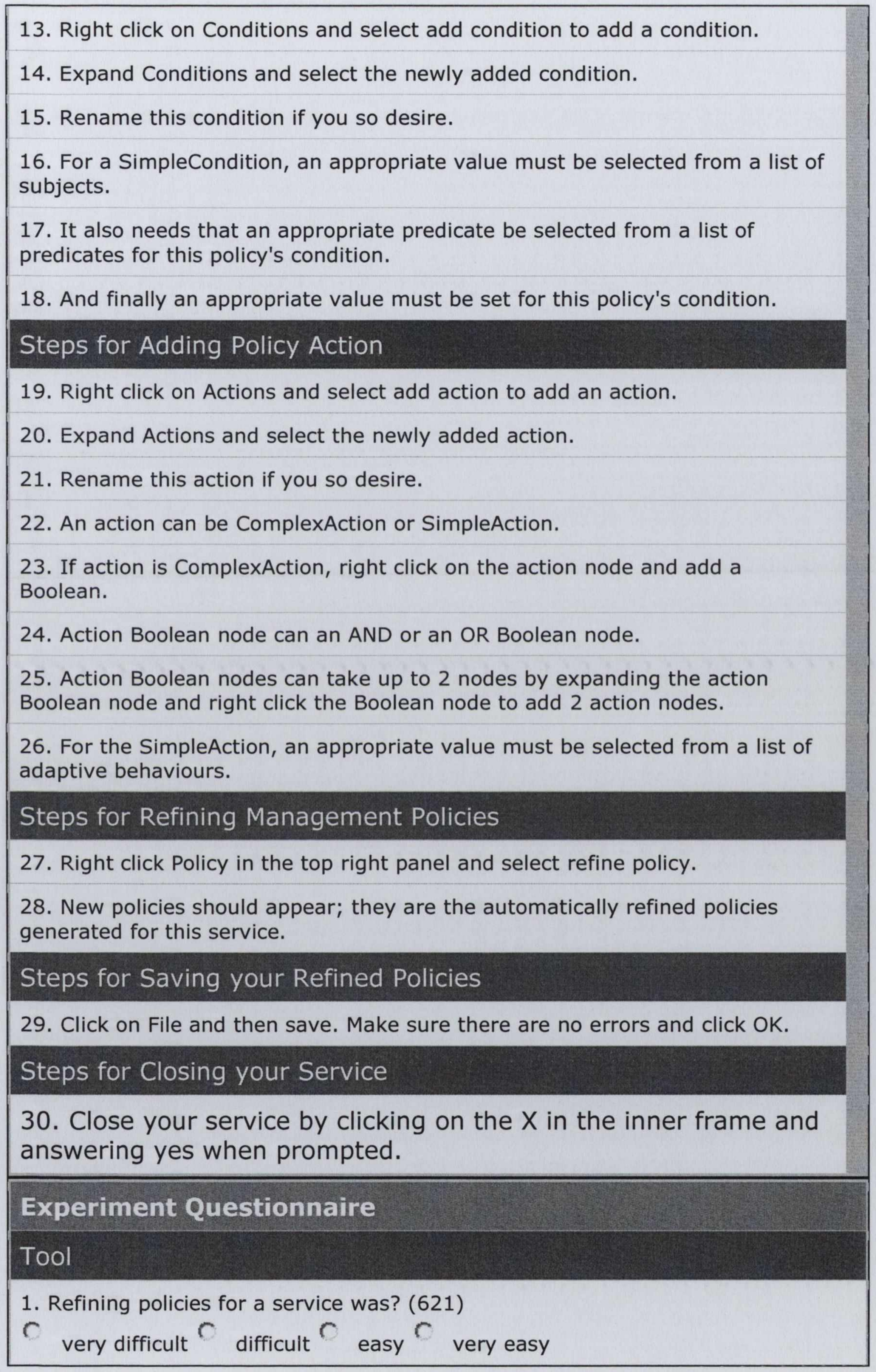
9. Right click on Events and select add event to add an event.

10. Expand Events and select the newly added event.

11. Rename this event if you so desire.

12. For a SimpleEvent an appropriate value must be selected from a list of events.

Steps for Adding Policy Condition


13. Right click on Conditions and select add condition to add a condition.

14. Expand Conditions and select the newly added condition.

15. Rename this condition if you so desire.

16. For a SimpleCondition, an appropriate value must be selected from a list of subjects.

17. It also needs that an appropriate predicate be selected from a list of predicates for this policy's condition.

18. And finally an appropriate value must be set for this policy's condition.

Steps for Adding Policy Action

19. Right click on Actions and select add action to add an action.

20. Expand Actions and select the newly added action.

21. Rename this action if you so desire.

22. An action can be ComplexAction or SimpleAction.

23. If action is ComplexAction, right click on the action node and add a Boolean.

24. Action Boolean node can an AND or an OR Boolean node.

25. Action Boolean nodes can take up to 2 nodes by expanding the action Boolean node and right click the Boolean node to add 2 action nodes.

26. For the SimpleAction, an appropriate value must be selected from a list of adaptive behaviours.

Steps for Refining Management Policies

27. Right click Policy in the top right panel and select refine policy.

28. New policies should appear; they are the automatically refined policies generated for this service.

Steps for Saving your Refined Policies

29. Click on File and then save. Make sure there are no errors and click OK.

Steps for Closing your Service

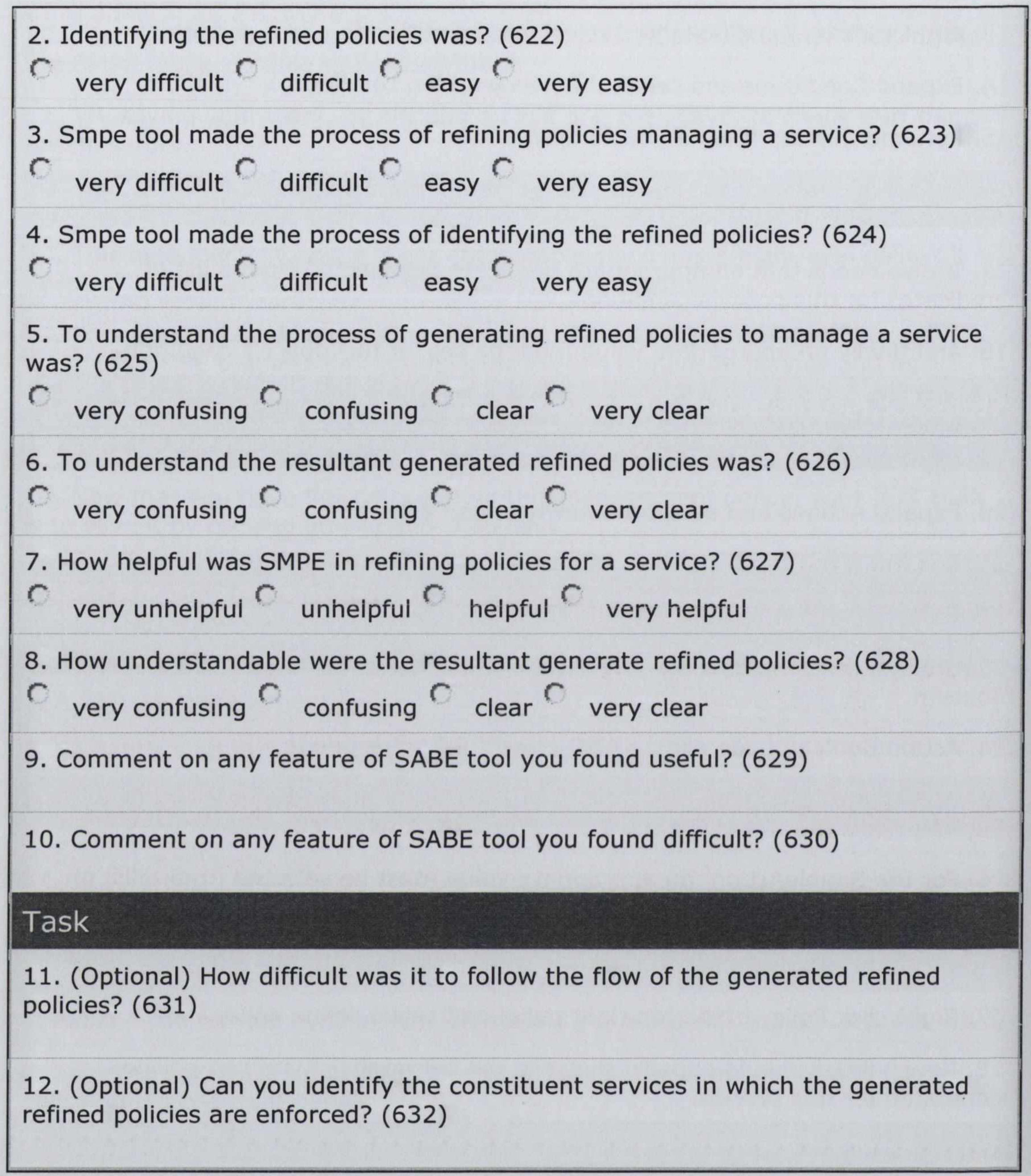
30. Close your service by clicking on the X in the inner frame and answering yes when prompted.

Experiment Questionnaire

Tool

1. Refining policies for a service was? (621)

○ very difficult ○ difficult ○ easy ○ very easy



2. Identifying the refined policies was? (622)

○ very difficult ○ difficult ○ easy ○ very easy

3. Smpe tool made the process of refining policies managing a service? (623)

○ very difficult ○ difficult ○ easy ○ very easy

4. Smpe tool made the process of identifying the refined policies? (624)

○ very difficult ○ difficult ○ easy ○ very easy

5. To understand the process of generating refined policies to manage a service was? (625)

○ very confusing ○ confusing ○ clear ○ very clear

6. To understand the resultant generated refined policies was? (626)

○ very confusing ○ confusing ○ clear ○ very clear

7. How helpful was SMPE in refining policies for a service? (627)

○ very unhelpful ○ unhelpful ○ helpful ○ very helpful

8. How understandable were the resultant generate refined policies? (628)

○ very confusing ○ confusing ○ clear ○ very clear

9. Comment on any feature of SABE tool you found useful? (629)

10. Comment on any feature of SABE tool you found difficult? (630)

Task

11. (Optional) How difficult was it to follow the flow of the generated refined policies? (631)

12. (Optional) Can you identify the constituent services in which the generated refined policies are enforced? (632)

Appendix D – SMPE Tool Usability Evaluation Results

The results of the pre-test and usability questionnaires for the three SMPE tool usability evaluation cycles. Analytical results of the Chi-square test performed over the three sets of usability questionnaire results.

Results of the SMPE Tool Usability Evaluation Cycle 1

Pre-Test

Users	A	B	C	D	E	F	Z[13]
401. How would you rate your knowledge of Policy?	3	2	4	4	2	4	3
402. Have you studied or read about policies?	2	2	4	4	2	4	3
403. Have you ever seen a policy (rule)?	2	1	4	4	1	4	3
404. Have you ever used policy?	2	1	4	4	2	3	3
405. Have you ever created a policy?	2	1	4	4	1	3	3

Results of the SMPE Tool Usability Evaluation Cycle 1

Users	A	B	C	D	E	F
421. Starting the SMPE tool was?	3	4	4	3	3	4
422. Loading the service was?	3	4	4	3	4	4
423. Viewing the details of the policy managing this service was?	3	4	4	3	3	3
424. Identifying what event triggers the policy was?	3	4	4	2	3	3
425. Identifying what condition the policy is evaluating was?	3	4	4	3	3	3
426. Identifying what adaptive behaviour is the policy is performing as its action was?	3	4	3	3	3	3
427. To understand the description of the policy managing this service was?	3	4	3	3	3	2

[13] This is the (rounded) average of the pre-test results, where 4 is very positive and 1 is very negative

428. To understand the description of the event that triggers this policy was?	3	4	4	2	2	3
429. To understand the description of the condition that must be satisfied for this policy was?	3	4	4	3	3	3
430. To understand the description of the action that is performed for this policy was?	3	4	4	3	2	3
431. Smpe tool made the process of browsing the description of the policy managing this service?	3	4	4	3	3	3
432. Smpe tool made the process of browsing the description of the event that triggers this policy?	3	4	4	3	3	3
433. Smpe tool made the process of browsing the description of the condition that must be satisfied for this policy?	3	4	4	3	3	3
434. Smpe tool made the process of browsing the description of the action that is performed for this policy?	3	4	4	3	3	3
435. When viewing management policies, would you rate SMPE tool as?	3	4	4	3	3	3
521. Creating a policy for managing the service was?	3	4	4	3	3	3
522. Creating an event that would trigger this policy was?	3	4	3	3	3	3
523. Creating a condition that must be satisfied for this policy was?	3	4	4	3	3	3
524. Creating an action that would be performed by this policy was?	3	4	3	3	4	3
525. To understand the process of creating a policy for managing the service was?	3	4	4	3	3	3
526. To understand the process of describing the event that would trigger this policy was?	3	4	3	3	3	3
527. To understand the process of describing the condition that must be satisfied for this policy was?	3	4	4	3	4	3
528. To understand the process of describing the action that would be performed by this policy was?	3	4	3	3	4	2
529. Smpe tool made the process of creating the policy for managing the service?	3	4	4	3	3	3
530. Smpe tool made the process of describing the event that would trigger this policy?	3	4	4	3	3	3
531. Smpe tool made the process of describing the condition that must be satisfied for this policy?	3	4	4	3	3	3
532. Smpe tool made the process of describing the action that would be performed by this policy?	3	4	4	3	3	2
533. When describing management policies, would you rate SMPE tool as?	3	4	3	3	3	3
621. Refining policies for a service was?	3	4	4	3	3	4
622. Identifying the refined policies was?	3	4	4	3	3	3
623. Identifying the events of the refined policies was?	3	4	4	3	3	3

624. Identifying the conditions of the refined policies was?	3	4	4	3	4	3
625. Identifying the actions of the refined policies was?	3	4	4	3	4	3
626. To understand the refining process of policies for a service was?	3	4	3	3	2	2
627. To understand the refined policies was?	3	4	3	3	3	2
628. To understand the events of the refined policies was?	3	4	4	3	3	2
629. To understand the conditions of the refined policies was?	3	4	4	3	3	2
630. To understand the actions of the refined policies was?	3	4	4	3	3	2
631. Smpe tool made the process of refining policies managing a service?	3	4	4	3	4	2
632. Smpe tool made the process of identifying the refined policies?	3	4	4	3	3	3
633. Smpe tool made the process of identifying the events of the refined policies?	3	4	4	3	3	3
634. Smpe tool made the process of identifying the conditions of the refined policies?	3	4	4	3	3	3
635. Smpe tool made the process of identifying the actions of the refined policies?	3	4	4	3	3	3
636. How helpful was SMPE in refining policies for a service?	3	4	4	3	3	3
637. How efficient was SMPE in refining policies for a service?	3	4	4	3	3	3
638. How understandable were the resultant refined policies?	3	4	3	3	3	2
639. Was the actions of the refined policies consistent with the chosen adaptive behaviour?	3	4	3	3	3	3
640. When refining management policies, would you rate SMPE tool as?	3	4	4	3	3	3

Results of the SMPE Tool Usability Evaluation Cycle 2

Pre-Test

Users	A	B	C	D	E	F	G	H	I	J	K	L	Z[14]
401. How would you rate your knowledge of Policy?	2	2	3	2	2	2	2	2	3	3	2	2	2
402. Have you studied or read about policies?	2	2	3	2	2	2	2	2	2	3	2	3	2

[14] This is the (rounded) average of the pre-test results, where 4 is very positive and 1 is very negative

403. Have you seen a policy rule before?	1	1	2	2	2	2	2	3	3	4	2	3	2
404. Have you used policy before?	1	1	1	1	2	2	1	3	2	4	1	2	2
405. Have you created a policy before?	1	1	1	1	1	1	1	3	1	4	1	2	2

Results of the SMPE Tool Usability Evaluation Cycle 2

Task-related Questions

Users	A	B	C	D	E	F	G	H	I	J	K	L
421. What is the event triggering the management policy ServiceAdapt1Policy2?	1	1	1	1	1	1	1	1	1	1	1	1
422. What is the action the management policy ServiceAdapt1Policy2 will perform?	1	1	1	1	1	1	1	1	0	1	1	1
423. What is the condition the management policy ServiceAdapt1Policy2 must first satisfy?	1	1	1	1	1	1	1	1	1	1	1	1

Results of the SMPE Tool Usability Evaluation Cycle 2

Questionnaire

Users	A	B	C	D	E	F	G	H	I	J	K	L
424. Starting the SMPE tool was?	4	1	4	4	3	3	4	4	3	3	3	3
425. Loading the service was?	4	4	4	4	3	3	4	4	3	3	3	3
426. Viewing the details of the policy managing this service was?	4	4	4	4	3	3	4	3	3	3	3	3
427. Identifying what event triggers the policy was?	4	4	3	4	3	3	4	3	3	2	2	3
428. Identifying what condition the policy is evaluating was?	4	4	4	4	3	3	4	3	3	3	3	3
429. Identifying what adaptive behaviour is the policy is performing as its action was?	4	4	3	3	3	3	4	3	3	3	3	3
430. Smpe tool made the process of browsing the description of the policy managing this service?	3	4	3	4	3	3	4	3	3	3	3	3
431. Smpe tool made the process of browsing the description of the event that triggers this policy?	3	4	4	4	3	3	4	3	3	3	3	3
432. Smpe tool made the process of browsing the description of the condition that must be satisfied for this policy?	3	4	4	4	3	3	4	3	3	3	3	3

433. Smpe tool made the process of browsing the description of the action that is performed for this policy?	4	4	3	4	3	3	4	3	3	3	3	3
434. To understand the description of the policy managing this service using SMPE tool was?	4	3	3	3	3	3	4	3	3	3	2	3
435. To understand the description of the event that triggers this policy using SMPE tool was?	4	3	3	3	3	3	4	3	3	3	2	3
436. To understand the description of the condition that must be satisfied for this policy using SMPE tool was?	3	3	3	3	3	3	3	3	3	3	2	3
437. To understand the description of the action that is performed for this policy using SMPE tool was?	3	3	3	3	3	3	4	3	3	3	2	3
521. Creating a policy for managing the service was?	3	3	4	3	3	3	4	3	3	3	3	3
522. Creating an event that would trigger this policy was?	3	3	3	3	3	3	4	3	3	2	3	3
523. Creating a condition that must be satisfied for this policy was?	2	3	4	3	3	3	4	3	3	2	3	3
524. Creating an action that would be performed by this policy was?	2	2	3	3	3	3	4	3	3	3	3	3
525. Smpe tool made the process of creating the policy for managing the service?	3	3	4	3	3	3	4	3	3	2	3	3
526. Smpe tool made the process of describing the event that would trigger this policy?	3	3	3	3	3	3	4	3	3	2	3	3
527. Smpe tool made the process of describing the condition that must be satisfied for this policy?	3	3	3	3	3	3	4	3	3	2	3	3
528. Smpe tool made the process of describing the action that would be performed by this policy?	3	2	4	3	3	3	4	3	3	3	3	3
529. To understand the process of creating a policy for managing the service using SMPE tool was?	3	3	3	3	3	3	4	3	3	2	3	3
530. To understand the process of describing the event that would trigger this policy using SMPE tool was?	3	3	4	3	3	3	4	3	3	2	3	3
531. To understand the process of describing the condition that must be satisfied for this policy using SMPE tool was?	3	3	3	3	3	3	4	3	3	2	3	3
532. To understand the process of describing the action that would be performed by this policy using SMPE tool was?	3	3	3	3	3	3	4	3	3	3	3	3
621. Refining policies for a service was?	3	4	3	3	3	3	4	4	3	3	3	3
622. Identifying the refined policies was?	3	3	4	3	2	3	4	3	3	3	3	3
623. Smpe tool made the process of refining policies managing a service?	3	4	4	3	2	3	4	3	3	3	4	3
624. Smpe tool made the process of identifying the refined policies?	3	3	4	3	3	3	4	3	3	3	3	3

625. To understand the process of generating refined policies to manage a service was?	3	3	2	4	2	3	3	3	3	3	3	3
626. To understand the resultant generated refined policies was?	3	3	3	3	2	3	3	3	3	3	3	3
627. How helpful was SMPE in refining policies for a service?	3	3	4	3	2	3	4	3	3	3	3	3
628. How understandable were the resultant generate refined policies?	3	3	4	3	2	3	3	3	3	3	3	3

Results of the SMPE Tool Usability Evaluation Cycle 3 Pre-Test

Users	A	B	C	D	E	F	G	H	I	J	K	L	Z[15]
401. How would you rate your knowledge of Policy?	2	2	3	2	2	2	2	2	3	3	2	2	2
402. Have you studied or read about policies?	2	2	3	2	2	2	2	2	2	3	2	3	2
403. Have you seen a policy rule before?	1	1	2	2	2	2	2	3	3	4	2	3	2
404. Have you used policy before?	1	1	1	1	2	2	1	3	2	4	1	2	2
405. Have you created a policy before?	1	1	1	1	1	1	1	3	1	4	1	2	2

Results of the SMPE Tool Usability Evaluation Cycle 3 Task-related Questions

Users	A	B	C	D	E	F	G	H	I	J	K	L
421. What is the event triggering the management policy ServiceAdapt1Policy2?	1	0	1	0	1	1	1	1	1	1	1	1
422. What is the action the management policy ServiceAdapt1Policy2 will perform?	1	0	1	1	1	1	1	1	1	1	1	1
423. What is the condition the management policy ServiceAdapt1Policy2 must first satisfy?	1	0	1	1	1	1	1	1	1	1	1	1

[15] This is the (rounded) average of the pre-test results, where 4 is very positive and 1 is very negative

Results of the SMPE Tool Usability Evaluation Cycle 3 Questionnaire

Users	A	B	C	D	E	F	G	H	I	J	K	L
424. Starting the SMPE tool was?	4	3	3	4	3	4	4	4	4	3	3	4
425. Loading the service was?	4	3	3	4	3	4	4	4	4	3	3	4
426. Viewing the details of the policy managing this service was?	4	3	3	4	2	4	3	4	3	3	3	4
427. Identifying what event triggers the policy was?	4	3	3	3	3	3	3	4	3	3	2	3
428. Identifying what condition the policy is evaluating was?	4	3	3	3	3	3	3	4	3	3	3	3
429. Identifying what adaptive behaviour is the policy is performing as its action was?	4	3	3	3	2	3	3	4	3	3	3	3
430. Smpe tool made the process of browsing the description of the policy managing this service?	4	3	3	3	3	3	2	4	3	3	3	3
431. Smpe tool made the process of browsing the description of the event that triggers this policy?	4	3	3	3	3	3	2	4	3	3	3	3
432. Smpe tool made the process of browsing the description of the condition that must be satisfied for this policy?	4	3	3	3	3	3	3	4	3	3	3	3
433. Smpe tool made the process of browsing the description of the action that is performed for this policy?	4	3	3	3	3	3	3	4	3	3	3	3
434. To understand the description of the policy managing this service using SMPE tool was?	3	3	3	3	2	3	3	4	3	3	2	3
435. To understand the description of the event that triggers this policy using SMPE tool was?	3	3	3	3	2	3	3	4	3	3	2	3
436. To understand the description of the condition that must be satisfied for this policy using SMPE tool was?	3	3	3	3	3	3	3	3	3	3	2	3
437. To understand the description of the action that is performed for this policy using SMPE tool was?	3	3	3	3	3	3	3	4	3	3	2	3
521. Creating a policy for managing the service was?	4	3	4	3	3	4	3	3	4	3	3	3
522. Creating an event that would trigger this policy was?	4	3	4	3	3	4	3	3	4	3	2	3
523. Creating a condition that must be satisfied for this policy was?	4	3	4	3	3	4	3	3	4	3	2	3
524. Creating an action that would be performed by this policy was?	4	3	4	3	3	4	3	3	4	3	3	3
525. Smpe tool made the process of creating the policy for managing the service?	4	3	4	4	3	4	3	3	4	3	2	3

526. Smpe tool made the process of describing the event that would trigger this policy?	4	3	4	4	3	4	3	3	4	3	2	3
527. Smpe tool made the process of describing the condition that must be satisfied for this policy?	4	3	4	3	3	4	3	3	4	3	2	3
528. Smpe tool made the process of describing the action that would be performed by this policy?	4	3	4	3	3	4	3	3	4	3	3	3
529. To understand the process of creating a policy for managing the service using SMPE tool was?	3	3	4	3	3	4	2	3	4	3	2	3
530. To understand the process of describing the event that would trigger this policy using SMPE tool was?	3	3	4	3	3	4	2	3	4	3	2	3
531. To understand the process of describing the condition that must be satisfied for this policy using SMPE tool was?	3	3	4	3	3	4	2	3	4	3	2	3
532. To understand the process of describing the action that would be performed by this policy using SMPE tool was?	3	3	4	3	3	4	2	3	4	3	3	3
621. Refining policies for a service was?	4	3	2	3	3	3	3	3	3	4	3	3
622. Identifying the refined policies was?	3	3	2	3	3	3	3	3	3	4	3	3
623. Smpe tool made the process of refining policies managing a service?	4	3	4	3	3	3	3	4	3	4	3	3
624. Smpe tool made the process of identifying the refined policies?	4	3	4	3	3	3	3	4	3	4	3	3
625. To understand the process of generating refined policies to manage a service was?	3	3	3	3	3	3	3	3	3	3	3	3
626. To understand the resultant generated refined policies was?	3	3	3	3	2	3	3	3	3	3	2	3
627. How helpful was SMPE in refining policies for a service?	4	3	3	3	4	3	3	3	3	3	4	3
628. How understandable were the resultant generate refined policies?	3	3	3	3	2	3	3	3	3	3	2	3

Analysis of the SMPE Tool Usability Evaluations

SMPE Usability Evaluation Cycle 1

Usability Test 1	Easiness		Helpfulness		Comprehension	
Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*
Very Difficult	1	0	1	0	1	0
Difficult	2	1	2	0	2	4

Easy	3	16	3	16	3	13
Very Easy	4	7	4	8	4	7

Usability Test 2	**Easiness**		**Helpfulness**		**Comprehension**	
Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*
Very Difficult	1	0	1	0	1	0
Difficult	2	0	2	1	2	1
Easy	3	17	3	15	3	15
Very Easy	4	7	4	8	4	8

Usability Test 3	**Easiness**		**Helpfulness**		**Comprehension**	
Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*
Very Difficult	1	0	1	0	1	0
Difficult	2	0	2	1	2	6
Easy	3	17	3	22	3	16
Very Easy	4	13	4	13	4	8

SMPE Usability Evaluation Cycle 2

Usability Test 1	**Easiness**		**Helpfulness**		**Comprehension**	
Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*
Very Difficult	1	0	1	0	1	0
Difficult	2	2	2	0	2	4
Easy	3	29	3	33	3	39
Very Easy	4	17	4	15	4	5

Usability Test 2	Easiness		Helpfulness		Comprehension	
Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*
Very Difficult	1	0	1	0	1	0
Difficult	2	5	2	4	2	3
Easy	3	37	3	38	3	40
Very Easy	4	6	4	6	4	5

Usability Test 3	Easiness		Helpfulness		Comprehension	
Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*
Very Difficult	1	0	1	0	1	0
Difficult	2	1	2	1	2	3
Easy	3	18	3	17	3	20
Very Easy	4	5	4	6	4	1

SMPE Usability Evaluation Cycle 3

Usability Test 1	Easiness		Helpfulness		Comprehension	
Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*
Very Difficult	1	0	1	0	1	0
Difficult	2	3	2	2	2	6
Easy	3	34	3	38	3	39
Very Easy	4	11	4	8	4	3

Usability Test 2	Easiness		Helpfulness		Comprehension	
Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*

Very Difficult	1	0	1	0	1	0
Difficult	2	2	2	3	2	7
Easy	3	30	3	27	3	29
Very Easy	4	16	4	18	4	12

Usability Test 3	**Easiness**		**Helpfulness**		**Comprehension**	
Meaning	*Bin*	*Freq*	*Bin*	*Freq*	*Bin*	*Freq*
Very Difficult	1	0	1	0	1	0
Difficult	2	2	2	0	2	2
Easy	3	19	3	16	3	22
Very Easy	4	3	4	8	4	0

SMPE Comparison of Usability Evaluation Cycles

Usability Test 1	**Easiness**			
Observed	**A**	**B**	**C**	**Total**
Easy	23	46	*45*	114
Difficult	1	2	*3*	6
	24	48	*48*	120
Expected	**A**	**B**	**C**	
Easy	22.8	45.6	*45.6*	
Difficult	1.2	2.4	*2.4*	
Null Hypothesis	**Observed**	**Expected**	**Result**	
P-value	0.87671	0.05	***TRUE***	

Usability Test 1	**Helpfulness**			
Observed	**A**	**B**	**C**	**Total**
Easy	24	48	*46*	118
Difficult	0	0	*2*	2
	24	48	*48*	120
Expected	**A**	**B**	**C**	
Easy	23.6	47.2	*47.2*	
Difficult	0.4	0.8	*0.8*	
Null Hypothesis	**Observed**	**Expected**	**Result**	
P-value	0.217529	0.05	***TRUE***	

Usability Test 1	**Comprehension**			
Observed	**A**	**B**	**C**	**Total**
Easy	20	44	*42*	106
Difficult	4	4	*6*	14
	24	48	*48*	120
Expected	**A**	**B**	**C**	
Easy	21.2	42.4	*42.4*	
Difficult	2.8	5.6	*5.6*	
Null Hypothesis	**Observed**	**Expected**	**Result**	
P-value	0.567771	0.05	***TRUE***	

Usability Test 2	**Easiness**			
Observed	**A**	**B**	**C**	**Total**
Easy	24	43	*46*	113

Difficult	0	5	*2*	7
	24	48	*48*	120
Expected	**A**	**B**	**C**	
Easy	22.6	45.2	*45.2*	
Difficult	1.4	2.8	*2.8*	
Null Hypothesis	**Observed**	**Expected**	**Result**	
P-value	0.168208	0.05	***TRUE***	

Usability Test 2		**Helpfulness**		
Observed	**A**	**B**	**C**	**Total**
Easy	23	44	*45*	112
Difficult	1	4	*3*	8
	24	48	*48*	120
Expected	**A**	**B**	**C**	
Easy	22.4	44.8	*44.8*	
Difficult	1.6	3.2	*3.2*	
Null Hypothesis	**Observed**	**Expected**	**Result**	
P-value	0.791065	0.05	***TRUE***	

Usability Test 2		**Comprehension**		
Observed	**A**	**B**	**C**	**Total**
Easy	23	45	*41*	109
Difficult	1	3	*7*	11
	24	48	*48*	120
Expected	**A**	**B**	**C**	

Easy	21.8	43.6	*43.6*	
Difficult	2.2	4.4	*4.4*	
Null Hypothesis	**Observed**	**Expected**	**Result**	
P-value	0.234287	0.05	***TRUE***	

Usability Test 3	**Easiness**			
Observed	**A**	**B**	**C**	**Total**
Easy	30	23	*22*	75
Difficult	0	1	*2*	3
	30	24	*24*	78
Expected	**A**	**B**	**C**	
Easy	28.84615	23.07692	*23.07692*	
Difficult	1.153846	0.923077	*0.923077*	
Null Hypothesis	**Observed**	**Expected**	**Result**	
P-value	0.284601	0.05	***TRUE***	

Usability Test 3	**Helpfulness**			
Observed	**A**	**B**	**C**	**Total**
Easy	35	23	*24*	82
Difficult	1	1	*0*	2
	36	24	*24*	84
Expected	**A**	**B**	**C**	
Easy	35.14286	23.42857	*23.42857143*	
Difficult	0.857143	0.571429	*0.571428571*	
Null Hypothesis	**Observed**	**Expected**	**Result**	

P-value	0.625307	0.05	***TRUE***	

Usability Test 3	**Comprehension**			
Observed	**A**	**B**	**C**	**Total**
Easy	24	21	*22*	67
Difficult	6	3	*2*	11
	30	24	*24*	78
Expected	**A**	**B**	**C**	
Easy	25.76923	20.61538	*20.61538*	
Difficult	4.230769	3.384615	*3.384615*	
Null Hypothesis	**Observed**	**Expected**	**Result**	
P-value	0.455745	0.05	***TRUE***	

Appendix E – Ontology Models

The Ontology models for the Finite State Machine and Obligation Policy used by the SABE and SMPE tools.

FSM Ontology Model

```
<?xml version="1.0" encoding="UTF-8" ?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns="http://www.kdeg.cs.tcd.ie/FiniteStateMachine.owl#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.kdeg.cs.tcd.ie/FiniteStateMachine.owl">
  <owl:Ontology rdf:about="">
  </owl:Ontology>
  <owl:Class rdf:ID="Transition">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#transitionsOf"/>
        </owl:onProperty>
        <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#guard"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#effect"/>
        </owl:onProperty>
        <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#target"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
        <owl:ObjectProperty rdf:about="#source"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#internalTransitionOf"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class rdf:ID="ModelElement"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#trigger"/>
      </owl:onProperty>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Guard">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#guardOf"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf rdf:resource="#ModelElement"/>
</owl:Class>
<owl:Class rdf:ID="FinalState">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#State"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ChangeEvent">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Event"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="StateMachine">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#top"/>
      </owl:onProperty>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
```

```
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#context"/>
        </owl:onProperty>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#ModelElement"/>
  </owl:Class>
  <owl:Class rdf:ID="SubmachineState">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#submachine"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Class rdf:about="#CompositeState"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="SimpleState">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#State"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="SubState">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#StateVertex"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Procedure">
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#effectOf"/>
        </owl:onProperty>
        <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="SignalEvent">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Event"/>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#signal"/>
        </owl:onProperty>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:cardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
```

```
<owl:Class rdf:ID="Parameter">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#parameterOf"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="CallEvent">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Event"/>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:cardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#operation"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="StateVertex">
  <rdfs:subClassOf rdf:resource="#ModelElement"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#container"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="PseudoState">
  <rdfs:subClassOf rdf:resource="#StateVertex"/>
</owl:Class>
<owl:Class rdf:ID="State">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#doActivity"/>
      </owl:onProperty>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
      >1</owl:maxCardinality>
      <owl:onProperty>
        <owl:ObjectProperty rdf:about="#exit"/>
      </owl:onProperty>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
```

```
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#entry"/>
        </owl:onProperty>
        <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf rdf:resource="#StateVertex"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#topOf"/>
        </owl:onProperty>
        <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
        >1</owl:maxCardinality>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="TimeEvent">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Event"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Signal"/>
  <owl:Class rdf:ID="SynchState">
    <rdfs:subClassOf rdf:resource="#StateVertex"/>
  </owl:Class>
  <owl:Class rdf:ID="CompositeState">
    <rdfs:subClassOf rdf:resource="#State"/>
  </owl:Class>
  <owl:Class rdf:ID="Operation"/>
  <owl:Class rdf:ID="Event">
    <rdfs:subClassOf rdf:resource="#ModelElement"/>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="transitionsOf">
    <rdfs:domain rdf:resource="#Transition"/>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:about="#transitions"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="submachineOf">
    <rdfs:range rdf:resource="#SubmachineState"/>
    <rdfs:domain rdf:resource="#StateMachine"/>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:about="#submachine"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="internalTransitionOf">
    <rdfs:domain rdf:resource="#Transition"/>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:about="#internalTransition"/>
    </owl:inverseOf>
    <rdfs:range rdf:resource="#State"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="doActivity">
    <rdfs:range rdf:resource="#Procedure"/>
    <rdfs:domain rdf:resource="#State"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="parameter">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#parameterOf"/>
  </owl:inverseOf>
  <rdfs:range rdf:resource="#Parameter"/>
  <rdfs:domain rdf:resource="#Event"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="subvertex">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#container"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#CompositeState"/>
  <rdfs:range rdf:resource="#StateVertex"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="exit">
  <rdfs:range rdf:resource="#Procedure"/>
  <rdfs:domain rdf:resource="#State"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#TransitiveProperty"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="submachine">
  <owl:inverseOf rdf:resource="#submachineOf"/>
  <rdfs:range rdf:resource="#StateMachine"/>
  <rdfs:domain rdf:resource="#SubmachineState"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="target">
  <rdfs:domain rdf:resource="#Transition"/>
  <rdfs:range rdf:resource="#StateVertex"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#incoming"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="guardOf">
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#guard"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#Guard"/>
  <rdfs:range rdf:resource="#Transition"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="guard">
  <owl:inverseOf rdf:resource="#guardOf"/>
  <rdfs:range rdf:resource="#Guard"/>
  <rdfs:domain rdf:resource="#Transition"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="deferrableEvent">
  <rdfs:range rdf:resource="#Event"/>
  <rdfs:domain rdf:resource="#State"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="triggerOf">
  <rdfs:range rdf:resource="#Transition"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#trigger"/>
  </owl:inverseOf>
  <rdfs:domain rdf:resource="#Event"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="topOf">
  <rdfs:domain rdf:resource="#State"/>
  <rdfs:range rdf:resource="#StateMachine"/>
  <owl:inverseOf rdf:resource="#top"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="effect">
  <rdfs:range rdf:resource="#Procedure"/>
```

```
    <rdfs:domain rdf:resource="#Transition"/>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:about="#effectOf"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="trigger">
    <rdfs:domain rdf:resource="#Transition"/>
    <owl:inverseOf rdf:resource="#triggerOf"/>
    <rdfs:range rdf:resource="#Event"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="internalTransition">
    <owl:inverseOf rdf:resource="#internalTransitionOf"/>
    <rdfs:range rdf:resource="#Transition"/>
    <rdfs:domain rdf:resource="#State"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="operationOf">
    <owl:inverseOf>
      <owl:ObjectProperty rdf:about="#operation"/>
    </owl:inverseOf>
    <rdfs:domain rdf:resource="#Operation"/>
    <rdfs:range rdf:resource="#CallEvent"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="effectOf">
    <rdfs:range rdf:resource="#Transition"/>
    <rdfs:domain rdf:resource="#Procedure"/>
    <owl:inverseOf rdf:resource="#effect"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="entry">
    <rdfs:domain rdf:resource="#State"/>
    <rdfs:range rdf:resource="#Procedure"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="incoming">
    <rdfs:domain rdf:resource="#StateVertex"/>
    <rdfs:range rdf:resource="#Transition"/>
    <owl:inverseOf rdf:resource="#target"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="container">
    <rdfs:range rdf:resource="#CompositeState"/>
    <rdfs:domain rdf:resource="#StateVertex"/>
    <owl:inverseOf rdf:resource="#subvertex"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="parameterOf">
    <rdfs:domain rdf:resource="#Parameter"/>
    <owl:inverseOf rdf:resource="#parameter"/>
    <rdfs:range rdf:resource="#Event"/>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="outgoing">
    <rdfs:domain rdf:resource="#StateVertex"/>
    <rdfs:range rdf:resource="#Transition"/>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:about="#source"/>
    </owl:inverseOf>
  </owl:ObjectProperty>
  <owl:ObjectProperty rdf:ID="signal">
    <rdfs:range rdf:resource="#Signal"/>
    <owl:inverseOf>
      <owl:ObjectProperty rdf:about="#signalOf"/>
    </owl:inverseOf>
    <rdfs:domain rdf:resource="#SignalEvent"/>
  </owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="top">
  <rdfs:range rdf:resource="#State"/>
  <rdfs:domain rdf:resource="#StateMachine"/>
  <owl:inverseOf rdf:resource="#topOf"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="behaviour">
  <rdfs:domain rdf:resource="#ModelElement"/>
  <owl:inverseOf>
    <owl:ObjectProperty rdf:about="#context"/>
  </owl:inverseOf>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="context">
  <rdfs:domain rdf:resource="#StateMachine"/>
  <owl:inverseOf rdf:resource="#behaviour"/>
  <rdfs:range rdf:resource="#ModelElement"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="signalOf">
  <rdfs:domain rdf:resource="#Signal"/>
  <rdfs:range rdf:resource="#SignalEvent"/>
  <owl:inverseOf rdf:resource="#signal"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="transitions">
  <rdfs:range rdf:resource="#Transition"/>
  <owl:inverseOf rdf:resource="#transitionsOf"/>
  <rdfs:domain rdf:resource="#StateMachine"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="source">
  <rdfs:range rdf:resource="#StateVertex"/>
  <owl:inverseOf rdf:resource="#outgoing"/>
  <rdfs:domain rdf:resource="#Transition"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="operation">
  <rdfs:range rdf:resource="#Operation"/>
  <rdfs:domain rdf:resource="#CallEvent"/>
  <owl:inverseOf rdf:resource="#operationOf"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="changeExpression">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
  <rdfs:domain rdf:resource="#ChangeEvent"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="kind">
  <rdfs:range>
    <owl:DataRange>
      <owl:oneOf rdf:parseType="Resource">
        <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >initial</rdf:first>
        <rdf:rest rdf:parseType="Resource">
          <rdf:rest rdf:parseType="Resource">
            <rdf:rest rdf:parseType="Resource">
              <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
              >join</rdf:first>
              <rdf:rest rdf:parseType="Resource">
                <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >fork</rdf:first>
                <rdf:rest rdf:parseType="Resource">
                  <rdf:rest rdf:parseType="Resource">
                    <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                    >choice</rdf:first>
                    <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
                  </rdf:rest>
                <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
                >junction</rdf:first>
              </rdf:rest>
            </rdf:rest>
          </rdf:rest>
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >shallowHistory</rdf:first>
        </rdf:rest>
        <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
        >deepHistory</rdf:first>
      </rdf:rest>
    </owl:oneOf>
  </owl:DataRange>
 </rdfs:range>
 <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty"/>
 <rdfs:domain rdf:resource="#PseudoState"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="isRegion">
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
 <rdfs:domain rdf:resource="#CompositeState"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="expression">
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
 <rdfs:domain rdf:resource="#Guard"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="when">
 <rdfs:domain rdf:resource="#TimeEvent"/>
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#dateTime"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="bound">
 <rdfs:domain rdf:resource="#SynchState"/>
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#positiveInteger"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="isConcurrent">
 <rdfs:domain rdf:resource="#CompositeState"/>
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#boolean"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="referenceState">
 <rdfs:domain rdf:resource="#SubState"/>
 <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#anyURI"/>
</owl:DatatypeProperty>
</rdf:RDF>
```

Obligation Policy Ontology Model

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE uridef [
 <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
 <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
 <!ENTITY owl "http://www.w3.org/2002/07/owl">
 <!ENTITY policy "http://kdeg.cs.tcd.ie/Policy.owl">
 <!ENTITY DEFAULT "http://kdeg.cs.tcd.ie/Policy.owl">
 <!ENTITY THIS "http://kdeg.cs.tcd.ie/Policy.owl">
]>
<rdf:RDF
 xmlns:rdf = "&rdf;#"
 xmlns:rdfs = "&rdfs;#"
 xmlns:owl = "&owl;#"
 xmlns =     "&policy;#">
<owl:Ontology rdf:about="">
 <owl:versionInfo>
  1.0
 </owl:versionInfo>
 <rdfs:comment>
  OWL ontology for obligation policy.
 </rdfs:comment>
</owl:Ontology>
<!-- Policy -->
<owl:Class rdf:ID="Policy">
 <rdfs:label>Policy</rdfs:label>
 <rdfs:comment>Obligation Policy</rdfs:comment>
</owl:Class>
<owl:DatatypeProperty rdf:ID="subject">
 <rdfs:domain rdf:resource="#Policy"/>
 <rdfs:range rdf:resource="&xsd;#anyURI"/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID="target">
 <rdfs:domain rdf:resource="#Policy"/>
 <rdfs:range rdf:resource="&xsd;#anyURI"/>
</owl:DatatypeProperty>
<!-- Event -->
<owl:ObjectProperty rdf:ID="event">
 <rdfs:domain rdf:resource="#Policy"/>
 <rdfs:range rdf:resource="#Event"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="Event">
 <owl:unionOf rdf:parseType="Collection">
  <owl:Class rdf:about="#SimpleEvent"/>
  <owl:Class rdf:about="#ComplexEvent"/>
 </owl:unionOf>
</owl:Class>
<owl:Class rdf:ID="SimpleEvent">
 <rdfs:subClassOf rdf:resource="#Event"/>
 <rdfs:label>Simple Event</rdfs:label>
 <rdfs:comment>Simple Event</rdfs:comment>
</owl:Class>
<owl:DatatypeProperty rdf:ID="value">
 <rdfs:comment>
  Value of a policy event aspect.
 </rdfs:comment>
 <rdfs:domain rdf:resource="#SimpleEvent"/>
 <rdfs:range rdf:resource="&xsd;#anyURI"/>
</owl:ObjectProperty>
<!-- Condition -->
<owl:ObjectProperty rdf:ID="condition">
 <rdfs:domain rdf:resource="#Policy"/>
 <rdfs:range rdf:resource="#Condition"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="Condition">
 <owl:unionOf rdf:parseType="Collection">
  <owl:Class rdf:about="#SimpleCondition"/>
  <owl:Class rdf:about="#ComplexCondition"/>
 </owl:unionOf>
</owl:Class>
<owl:Class rdf:ID="SimpleCondition">
 <rdfs:subClassOf rdf:resource="#Condition"/>
 <rdfs:label>SimpleCondition</rdfs:label>
 <rdfs:comment>Simple Condition</rdfs:comment>
```

```
</owl:Class>
<owl:DatatypeProperty rdf:ID="subject">
 <rdfs:comment>
  Subject of a policy condition aspect.
 </rdfs:comment>
 <rdfs:domain rdf:resource="#SimpleCondition"/>
 <rdfs:range rdf:resource="&xsd;#anyURI"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="predicate">
 <rdfs:comment>
  Predicate of a policy condition aspect.
 </rdfs:comment>
 <rdfs:domain rdf:resource="#SimpleCondition"/>
 <rdfs:range rdf:resource="#Predicate"/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID="value">
 <rdfs:comment>
  Value of a policy condition aspect.
 </rdfs:comment>
 <rdfs:domain rdf:resource="#SimpleCondition"/>
 <rdfs:range rdf:resource="&xsd;#anyURI"/>
</owl:ObjectProperty>
<owl:Class rdf:ID="Predicate">
 <rdfs:label>Predicate</rdfs:label>
 <rdfs:comment>Predicate</rdfs:comment>
</owl:Class>
<Predicate rdf:ID="equal"/>
<Predicate rdf:ID="inequal"/>
<Predicate rdf:ID="greater"/>
<Predicate rdf:ID="less"/>
<!-- Action -->
<owl:ObjectProperty rdf:ID="action">
 <rdfs:domain rdf:resource="#Policy"/>
 <rdfs:range rdf:resource="#Action"/>
</owl:DatatypeProperty>
<owl:Class rdf:ID="Action">
 <owl:unionOf rdf:parseType="Collection">
  <owl:Class rdf:about="#SimpleAction"/>
  <owl:Class rdf:about="#ComplexAction"/>
 </owl:unionOf>
</owl:Class>
<owl:Class rdf:ID="SimpleAction">
 <rdfs:subClassOf rdf:resource="#Action"/>
 <rdfs:label>SimpleAction</rdfs:label>
 <rdfs:comment>Simple Action</rdfs:comment>
</owl:Class>
<owl:DatatypeProperty rdf:ID="value">
 <rdfs:comment>
  Value of a policy action aspect.
 </rdfs:comment>
 <rdfs:domain rdf:resource="#SimpleAction"/>
 <rdfs:range rdf:resource="&xsd;#anyURI"/>
</owl:ObjectProperty>
<!-- Comples (Multiple) aspects -->
<owl:Class rdf:ID="ComplexEvent">
 <rdfs:subClassOf rdf:resource="#Event"/>
 <rdfs:label>Complex Event</rdfs:label>
 <rdfs:comment>Complex Event</rdfs:comment>
  <owl:Restriction>
   <owl:onProperty rdf:resource="&rdfs;#first"/>
```

```
    <owl:allValuesFrom rdf:resource="#PolicyAspect"/>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="&rdfs;#rest"/>
    <owl:allValuesFrom rdf:resource="#PolicyAspect"/>
   </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ComplexCondition">
  <rdfs:subClassOf rdf:resource="#Condition"/>
  <rdfs:label>ComplexCondition</rdfs:label>
  <rdfs:comment>Complex Condition</rdfs:comment>
   <owl:Restriction>
    <owl:onProperty rdf:resource="&rdfs;#first"/>
    <owl:allValuesFrom rdf:resource="#PolicyAspect"/>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="&rdfs;#rest"/>
    <owl:allValuesFrom rdf:resource="#PolicyAspect"/>
   </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="ComplexAction">
  <rdfs:subClassOf rdf:resource="#Action"/>
  <rdfs:label>ComplexAction</rdfs:label>
  <rdfs:comment>Complex Action</rdfs:comment>
   <owl:Restriction>
    <owl:onProperty rdf:resource="&rdfs;#first"/>
    <owl:allValuesFrom rdf:resource="#PolicyAspect"/>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="&rdfs;#rest"/>
    <owl:allValuesFrom rdf:resource="#PolicyAspect"/>
   </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="PolicyAspect">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#SimpleEvent"/>
    <owl:Class rdf:about="#SimpleCondition"/>
    <owl:Class rdf:about="#SimpleAction"/>
    <owl:Class rdf:about="#AndList"/>
    <owl:Class rdf:about="#OrList"/>
  </owl:unionOf>
</owl:Class>
<owl:Class rdf:ID="AndList">
<rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="&rdfs;#first"/>
    <owl:allValuesFrom rdf:resource="#PolicyAspect"/>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
```

```
    <owl:onProperty rdf:resource="&rdfs;#rest"/>
    <owl:allValuesFrom rdf:resource="#PolicyAspect"/>
   </owl:Restriction>
  </rdfs:subClassOf>
 </owl:Class>
 <owl:Class rdf:ID="OrList">
 <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="&rdfs;#first"/>
    <owl:allValuesFrom rdf:resource="#PolicyAspect"/>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="&rdfs;#rest"/>
    <owl:allValuesFrom rdf:resource="#PolicyAspect"/>
   </owl:Restriction>
  </rdfs:subClassOf>
 </owl:Class>
 </rdf:RDF>
```

Appendix F – Personalised Holiday Service Case Study

The artefacts produced for the Personalised Holiday service case study. The artefacts are the description of the composite service, expressed as OWL-S; the description of the adaptive behaviours, expressed as a FSM; the high level management policies; and auto-generated refined low level enforceable policies. Lastly, the runtime trace of the web service managed by the refined policies.

PersonalHoliday Service

```
<rdf:RDF
  xmlns="http://www.daml.org/services/owl-s/1.1/PersonalHolidayService.owl#"
  xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
  xmlns:thisgrounding="http://www.daml.org/services/owl-s/1.1/PersonalHolidayGrounding.owl#"
  xmlns:policy="http://kdeg.cs.tcd.ie/Policy.owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:thisprocess="http://www.daml.org/services/owl-s/1.1/PersonalHolidayProcess.owl#"
  xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
  xmlns:base="http://www.daml.org/services/owl-s/1.1/PersonalHolidayService.owl"
  xmlns:grounding="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:thisprofile="http://www.daml.org/services/owl-s/1.1/PersonalHolidayProfile.owl#"
  xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
 xml:base="http://www.daml.org/services/owl-s/1.1/PersonalHolidayService.owl">
 <owl:Ontology rdf:about="">
  <owl:imports rdf:resource="PersonalHolidayProfile.owl"/>
  <owl:imports rdf:resource="PersonalHolidayProcess.owl"/>
  <owl:imports rdf:resource="PersonalHolidayGrounding.owl"/>
  <owl:imports rdf:resource="Process.owl"/>
  <owl:imports rdf:resource="Service.owl"/>
  <owl:imports rdf:resource="Grounding.owl"/>
  <owl:versionInfo></owl:versionInfo>
  <owl:imports rdf:resource="Profile.owl"/>
  <rdfs:comment></rdfs:comment>
 </owl:Ontology>
 <service:Service rdf:ID="PersonalHolidayService">
  <service:describedBy rdf:resource="PersonalHolidayProcess.owl#PersonalHolidayService"/>
  <service:presents rdf:resource="PersonalHolidayProfile.owl#PersonalHolidayProfile"/>
  <service:supports rdf:resource="PersonalHolidayGrounding.owl#PersonalHolidayGrounding"/>
 </service:Service>
</rdf:RDF>
```

PersonalHoliday Process

```
<rdf:RDF
  xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
  xmlns:objList="http://www.daml.org/services/owl-s/1.1/generic/ObjectList.owl#"
  xmlns:policy="http://kdeg.cs.tcd.ie/Policy.owl#"
  xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
  xmlns:grounding="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:base="http://www.daml.org/services/owl-s/1.1/PersonalHolidayProcess.owl"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
  xmlns="http://www.daml.org/services/owl-s/1.1/PersonalHolidayProcess.owl#"
  xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
xml:base="http://www.daml.org/services/owl-s/1.1/PersonalHolidayProcess.owl">
<owl:Ontology rdf:about="">
  <owl:imports rdf:resource="Service.owl"/>
  <owl:imports rdf:resource="PersonalHolidayService.owl"/>
  <owl:imports rdf:resource="Grounding.owl"/>
  <owl:imports rdf:resource="Profile.owl"/>
  <owl:imports rdf:resource="PersonalHolidayProfile.owl"/>
  <owl:imports rdf:resource="PersonalHolidayGrounding.owl"/>
  <owl:versionInfo></owl:versionInfo>
  <rdfs:comment></rdfs:comment>
  <owl:imports rdf:resource="Process.owl"/>
</owl:Ontology>
<process:Input rdf:ID="HotelCost">
  <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
  >http://www.w3.org/2001/XMLSchema#double</process:parameterType>
</process:Input>
<process:AtomicProcess rdf:ID="PaymentService">
  <process:hasInput>
    <process:Output rdf:ID="Guestid">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
      <rdf:type rdf:resource="Process.owl#Input"/>
    </process:Output>
  </process:hasInput>
  <process:hasInput>
    <process:Input rdf:ID="Charge">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/2001/XMLSchema#double</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput>
    <process:Output rdf:ID="SalesTax">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/2001/XMLSchema#double</process:parameterType>
      <rdf:type rdf:resource="Process.owl#Input"/>
    </process:Output>
  </process:hasInput>
  <process:hasOutput>
    <process:Output rdf:ID="TotalCost">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/2001/XMLSchema#double</process:parameterType>
    </process:Output>
  </process:hasOutput>
  <process:hasOutput>
    <process:Output rdf:ID="Approved">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/2001/XMLSchema#boolean</process:parameterType>
    </process:Output>
  </process:hasOutput>
  <process:hasFiniteStateMachine rdf:resource="PaymentServiceFSM.owl#PaymentServiceFSM"/>
</process:AtomicProcess>
```

```
<process:AtomicProcess rdf:ID="FlightService">
  <process:hasOutput>
    <process:Output rdf:ID="Itinerary">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
    </process:Output>
  </process:hasOutput>
  <process:hasInput>
    <process:Input rdf:ID="Seats">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/2001/XMLSchema#int</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasOutput>
    <process:Output rdf:ID="Cost">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/2001/XMLSchema#double</process:parameterType>
    </process:Output>
  </process:hasOutput>
  <process:hasInput>
    <process:Input rdf:ID="Depart">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/2001/XMLSchema#date</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasOutput>
    <process:Output rdf:ID="FlightNumber">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
    </process:Output>
  </process:hasOutput>
  <process:hasInput>
    <process:Input rdf:ID="Origin">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasInput rdf:resource="#Guestid"/>
  <process:hasOutput>
    <process:Output rdf:ID="ArriveHome">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/2001/XMLSchema#date</process:parameterType>
    </process:Output>
  </process:hasOutput>
  <process:hasOutput>
    <process:Output rdf:ID="FlightClass">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
    </process:Output>
  </process:hasOutput>
  <process:hasInput>
    <process:Input rdf:ID="Return">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/2001/XMLSchema#date</process:parameterType>
    </process:Input>
  </process:hasInput>
  <process:hasOutput>
    <process:Output rdf:ID="ReservationCode">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
    </process:Output>
   </process:hasOutput>
   <process:hasFiniteStateMachine rdf:resource="FlightServiceFSM.owl#FlightServiceFSM"/>
   <process:hasInput>
    <process:Input rdf:ID="Destination">
     <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
     >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
    </process:Input>
   </process:hasInput>
   <process:hasOutput>
    <process:Output rdf:ID="ArriveDestination">
     <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
     >http://www.w3.org/2001/XMLSchema#date</process:parameterType>
    </process:Output>
   </process:hasOutput>
  </process:AtomicProcess>
  <process:Input rdf:ID="FlightCost">
   <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
   >http://www.w3.org/2001/XMLSchema#double</process:parameterType>
  </process:Input>
  <process:Output rdf:ID="HotelReservation">
   <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
   >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
  </process:Output>
  <process:Perform rdf:ID="HotelServicePerform">
   <process:hasDataFrom>
    <process:InputBinding>
     <process:valueSource>
      <process:ValueOf>
       <process:theVar>
        <process:Output rdf:ID="HotelAddress">
         <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
         >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
        </process:Output>
       </process:theVar>
       <process:fromProcess rdf:resource="#TheParentPerform"/>
      </process:ValueOf>
     </process:valueSource>
     <process:toParam rdf:resource="#HotelAddress"/>
    </process:InputBinding>
   </process:hasDataFrom>
   <process:hasDataFrom>
    <process:InputBinding>
     <process:valueSource>
      <process:ValueOf>
       <process:theVar>
        <process:Output rdf:ID="HotelFacilities">
         <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
         >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
        </process:Output>
       </process:theVar>
       <process:fromProcess rdf:resource="#TheParentPerform"/>
      </process:ValueOf>
     </process:valueSource>
     <process:toParam>
      <process:Output rdf:ID="Facilities">
       <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
       >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
      </process:Output>
     </process:toParam>
   </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
   <process:InputBinding>
    <process:valueSource>
     <process:ValueOf>
      <process:theVar rdf:resource="#HotelReservation"/>
      <process:fromProcess rdf:resource="#TheParentPerform"/>
     </process:ValueOf>
    </process:valueSource>
    <process:toParam rdf:resource="#ReservationCode"/>
   </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
   <process:InputBinding>
    <process:valueSource>
     <process:ValueOf>
      <process:theVar rdf:resource="#Guestid"/>
      <process:fromProcess rdf:resource="#TheParentPerform"/>
     </process:ValueOf>
    </process:valueSource>
    <process:toParam rdf:resource="#Guestid"/>
   </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
   <process:InputBinding>
    <process:valueSource>
     <process:ValueOf>
      <process:theVar>
       <process:Input rdf:ID="Persons">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/2001/XMLSchema#int</process:parameterType>
       </process:Input>
      </process:theVar>
      <process:fromProcess rdf:resource="#TheParentPerform"/>
     </process:ValueOf>
    </process:valueSource>
    <process:toParam>
     <process:Input rdf:ID="Rooms">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/2001/XMLSchema#int</process:parameterType>
     </process:Input>
    </process:toParam>
   </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
   <process:InputBinding>
    <process:valueSource>
     <process:ValueOf>
      <process:theVar rdf:resource="#ArriveDestination"/>
      <process:fromProcess>
       <process:Perform rdf:ID="FlightServicePerform">
        <process:process rdf:resource="#FlightService"/>
        <process:hasDataFrom>
         <process:InputBinding>
          <process:valueSource>
           <process:ValueOf>
            <process:theVar rdf:resource="#Return"/>
            <process:fromProcess rdf:resource="#TheParentPerform"/>
           </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#Return"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#FlightNumber"/>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#FlightNumber"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar>
            <process:Output rdf:ID="FlightItinerary">
              <process:parameterType rdf:datatype=
              "http://www.w3.org/2001/XMLSchema#anyURI"
              >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
            </process:Output>
          </process:theVar>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#Itinerary"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#Guestid"/>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#Guestid"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#Origin"/>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#Origin"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#Destination"/>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
```

```
            </process:ValueOf>
          </process:valueSource>
          <process:toParam rdf:resource="#Destination"/>
        </process:InputBinding>
      </process:hasDataFrom>
      <process:hasDataFrom>
        <process:InputBinding>
          <process:valueSource>
            <process:ValueOf>
              <process:theVar rdf:resource="#Depart"/>
              <process:fromProcess rdf:resource="#TheParentPerform"/>
            </process:ValueOf>
          </process:valueSource>
          <process:toParam rdf:resource="#Depart"/>
        </process:InputBinding>
      </process:hasDataFrom>
      <process:hasDataFrom>
        <process:InputBinding>
          <process:valueSource>
            <process:ValueOf>
              <process:theVar>
                <process:Output rdf:ID="FlightReservation">
                  <process:parameterType rdf:datatype=
                  "http://www.w3.org/2001/XMLSchema#anyURI"
                  >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
                </process:Output>
              </process:theVar>
              <process:fromProcess rdf:resource="#TheParentPerform"/>
            </process:ValueOf>
          </process:valueSource>
          <process:toParam rdf:resource="#ReservationCode"/>
        </process:InputBinding>
      </process:hasDataFrom>
      <process:hasDataFrom>
        <process:InputBinding>
          <process:valueSource>
            <process:ValueOf>
              <process:theVar rdf:resource="#FlightClass"/>
              <process:fromProcess rdf:resource="#TheParentPerform"/>
            </process:ValueOf>
          </process:valueSource>
          <process:toParam rdf:resource="#FlightClass"/>
        </process:InputBinding>
      </process:hasDataFrom>
      <process:hasDataFrom>
        <process:InputBinding>
          <process:valueSource>
            <process:ValueOf>
              <process:theVar rdf:resource="#Persons"/>
              <process:fromProcess rdf:resource="#TheParentPerform"/>
            </process:ValueOf>
          </process:valueSource>
          <process:toParam rdf:resource="#Seats"/>
        </process:InputBinding>
      </process:hasDataFrom>
    </process:Perform>
  </process:fromProcess>
 </process:ValueOf>
</process:valueSource>
<process:toParam>
```

```
      <process:Input rdf:ID="CheckinDate">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/2001/XMLSchema#date</process:parameterType>
      </process:Input>
    </process:toParam>
  </process:InputBinding>
</process:hasDataFrom>
<process:process>
  <process:AtomicProcess rdf:ID="HotelService">
    <process:hasInput rdf:resource="#CheckinDate"/>
    <process:hasOutput rdf:resource="#Facilities"/>
    <process:hasFiniteStateMachine rdf:resource="HotelServiceFSM.owl#HotelServiceFSM"/>
    <process:hasOutput>
      <process:Output rdf:ID="Brochure">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
      </process:Output>
    </process:hasOutput>
    <process:hasOutput>
      <process:Output rdf:ID="HotelName">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
      </process:Output>
    </process:hasOutput>
    <process:hasOutput>
      <process:Output rdf:ID="HotelStars">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/2001/XMLSchema#int</process:parameterType>
      </process:Output>
    </process:hasOutput>
    <process:hasOutput rdf:resource="#ReservationCode"/>
    <process:hasInput rdf:resource="#Rooms"/>
    <process:hasInput rdf:resource="#Destination"/>
    <process:hasOutput rdf:resource="#HotelAddress"/>
    <process:hasInput>
      <process:Input rdf:ID="CheckoutDate">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/2001/XMLSchema#date</process:parameterType>
      </process:Input>
    </process:hasInput>
    <process:hasInput rdf:resource="#Guestid"/>
    <process:hasOutput rdf:resource="#Cost"/>
  </process:AtomicProcess>
</process:process>
<process:hasDataFrom>
  <process:InputBinding>
    <process:valueSource>
      <process:ValueOf>
        <process:theVar rdf:resource="#HotelName"/>
        <process:fromProcess rdf:resource="#TheParentPerform"/>
      </process:ValueOf>
    </process:valueSource>
    <process:toParam rdf:resource="#HotelName"/>
  </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
  <process:InputBinding>
    <process:valueSource>
      <process:ValueOf>
        <process:theVar>
```

```
          <process:Output rdf:ID="HotelBrochure">
           <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
           >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
          </process:Output>
        </process:theVar>
        <process:fromProcess rdf:resource="#TheParentPerform"/>
      </process:ValueOf>
    </process:valueSource>
    <process:toParam rdf:resource="#Brochure"/>
  </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
  <process:InputBinding>
    <process:valueSource>
      <process:ValueOf>
        <process:theVar rdf:resource="#HotelStars"/>
        <process:fromProcess rdf:resource="#TheParentPerform"/>
      </process:ValueOf>
    </process:valueSource>
    <process:toParam rdf:resource="#HotelStars"/>
  </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
  <process:InputBinding>
    <process:valueSource>
      <process:ValueOf>
        <process:theVar rdf:resource="#Destination"/>
        <process:fromProcess rdf:resource="#TheParentPerform"/>
      </process:ValueOf>
    </process:valueSource>
    <process:toParam rdf:resource="#Destination"/>
  </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
  <process:InputBinding>
    <process:valueSource>
      <process:ValueOf>
        <process:theVar rdf:resource="#Return"/>
        <process:fromProcess rdf:resource="#TheParentPerform"/>
      </process:ValueOf>
    </process:valueSource>
    <process:toParam rdf:resource="#CheckoutDate"/>
  </process:InputBinding>
</process:hasDataFrom>
</process:Perform>
<process:AtomicProcess rdf:ID="HolidayBillService">
  <process:hasInput rdf:resource="#FlightCost"/>
  <process:hasInput rdf:resource="#HotelCost"/>
  <process:hasOutput>
    <process:Output rdf:ID="HolidayCost">
      <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/2001/XMLSchema#double</process:parameterType>
    </process:Output>
  </process:hasOutput>
  <process:hasOutput rdf:resource="#SalesTax"/>
</process:AtomicProcess>
<process:Perform rdf:ID="HolidayBillServicePerform">
  <process:process rdf:resource="#HolidayBillService"/>
  <process:hasDataFrom>
    <process:InputBinding>
```

```
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#Cost"/>
            <process:fromProcess rdf:resource="#HotelServicePerform"/>
          </process:ValueOf>
        </process:valueSource>
        <process:toParam rdf:resource="#HotelCost"/>
      </process:InputBinding>
    </process:hasDataFrom>
    <process:hasDataFrom>
      <process:InputBinding>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#Cost"/>
            <process:fromProcess rdf:resource="#FlightServicePerform"/>
          </process:ValueOf>
        </process:valueSource>
        <process:toParam rdf:resource="#HotelCost"/>
      </process:InputBinding>
    </process:hasDataFrom>
    <process:hasDataFrom>
      <process:InputBinding>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#HolidayCost"/>
            <process:fromProcess rdf:resource="#TheParentPerform"/>
          </process:ValueOf>
        </process:valueSource>
        <process:toParam rdf:resource="#HolidayCost"/>
      </process:InputBinding>
    </process:hasDataFrom>
    <process:hasDataFrom>
      <process:InputBinding>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#SalesTax"/>
            <process:fromProcess rdf:resource="#TheParentPerform"/>
          </process:ValueOf>
        </process:valueSource>
        <process:toParam rdf:resource="#SalesTax"/>
      </process:InputBinding>
    </process:hasDataFrom>
  </process:Perform>
  <process:Perform rdf:ID="LoginServicePerform">
    <process:process>
      <process:AtomicProcess rdf:ID="LoginService">
        <process:hasInput>
          <process:Input rdf:ID="Username">
            <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
            >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
          </process:Input>
        </process:hasInput>
        <process:hasInput>
          <process:Input rdf:ID="Password">
            <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
            >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
          </process:Input>
        </process:hasInput>
        <process:hasInput>
          <process:Input rdf:ID="Membership">
          <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
        </process:Input>
      </process:hasInput>
      <process:hasOutput rdf:resource="#Guestid"/>
    </process:AtomicProcess>
  </process:process>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#Username"/>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#Username"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#Password"/>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#Password"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#Membership"/>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#Membership"/>
    </process:InputBinding>
  </process:hasDataFrom>
</process:Perform>
<process:Perform rdf:ID="PaymentServicePerform">
  <process:process rdf:resource="#PaymentService"/>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#Guestid"/>
          <process:fromProcess rdf:resource="#LoginServicePerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#Guestid"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#SalesTax"/>
          <process:fromProcess>
```

```
<process:Perform rdf:ID="HolidayPackageServicePerform">
 <process:hasDataFrom>
  <process:InputBinding>
   <process:valueSource>
    <process:ValueOf>
     <process:theVar rdf:resource="#FlightNumber"/>
     <process:fromProcess rdf:resource="#TheParentPerform"/>
    </process:ValueOf>
   </process:valueSource>
   <process:toParam rdf:resource="#FlightNumber"/>
  </process:InputBinding>
 </process:hasDataFrom>
 <process:hasDataFrom>
  <process:InputBinding>
   <process:valueSource>
    <process:ValueOf>
     <process:theVar rdf:resource="#FlightItinerary"/>
     <process:fromProcess rdf:resource="#TheParentPerform"/>
    </process:ValueOf>
   </process:valueSource>
   <process:toParam rdf:resource="#FlightItinerary"/>
  </process:InputBinding>
 </process:hasDataFrom>
 <process:hasDataFrom>
  <process:InputBinding>
   <process:valueSource>
    <process:ValueOf>
     <process:theVar rdf:resource="#HotelName"/>
     <process:fromProcess rdf:resource="#TheParentPerform"/>
    </process:ValueOf>
   </process:valueSource>
   <process:toParam rdf:resource="#HotelName"/>
  </process:InputBinding>
 </process:hasDataFrom>
 <process:hasDataFrom>
  <process:InputBinding>
   <process:valueSource>
    <process:ValueOf>
     <process:theVar rdf:resource="#FlightClass"/>
     <process:fromProcess rdf:resource="#TheParentPerform"/>
    </process:ValueOf>
   </process:valueSource>
   <process:toParam rdf:resource="#FlightClass"/>
  </process:InputBinding>
 </process:hasDataFrom>
 <process:hasDataFrom>
  <process:InputBinding>
   <process:valueSource>
    <process:ValueOf>
     <process:theVar rdf:resource="#HotelReservation"/>
     <process:fromProcess rdf:resource="#TheParentPerform"/>
    </process:ValueOf>
   </process:valueSource>
   <process:toParam rdf:resource="#HotelReservation"/>
  </process:InputBinding>
 </process:hasDataFrom>
 <process:hasDataFrom>
  <process:InputBinding>
   <process:valueSource>
    <process:ValueOf>
    <process:theVar rdf:resource="#HotelBrochure"/>
    <process:fromProcess rdf:resource="#TheParentPerform"/>
   </process:ValueOf>
  </process:valueSource>
  <process:toParam rdf:resource="#HotelBrochure"/>
 </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
 <process:InputBinding>
  <process:valueSource>
   <process:ValueOf>
    <process:theVar rdf:resource="#Depart"/>
    <process:fromProcess rdf:resource="#TheParentPerform"/>
   </process:ValueOf>
  </process:valueSource>
  <process:toParam rdf:resource="#Depart"/>
 </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
 <process:InputBinding>
  <process:valueSource>
   <process:ValueOf>
    <process:theVar rdf:resource="#HotelFacilities"/>
    <process:fromProcess rdf:resource="#TheParentPerform"/>
   </process:ValueOf>
  </process:valueSource>
  <process:toParam rdf:resource="#HotelFacilities"/>
 </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
 <process:InputBinding>
  <process:valueSource>
   <process:ValueOf>
    <process:theVar rdf:resource="#Destination"/>
    <process:fromProcess rdf:resource="#TheParentPerform"/>
   </process:ValueOf>
  </process:valueSource>
  <process:toParam rdf:resource="#Destination"/>
 </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
 <process:InputBinding>
  <process:valueSource>
   <process:ValueOf>
    <process:theVar rdf:resource="#HotelStars"/>
    <process:fromProcess rdf:resource="#TheParentPerform"/>
   </process:ValueOf>
  </process:valueSource>
  <process:toParam rdf:resource="#HotelStars"/>
 </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
 <process:InputBinding>
  <process:valueSource>
   <process:ValueOf>
    <process:theVar rdf:resource="#Persons"/>
    <process:fromProcess rdf:resource="#TheParentPerform"/>
   </process:ValueOf>
  </process:valueSource>
  <process:toParam rdf:resource="#Persons"/>
```

```
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#FlightReservation"/>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#FlightReservation"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#Origin"/>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#Origin"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#HotelAddress"/>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#HotelAddress"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#Return"/>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#Return"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:process>
    <process:CompositeProcess rdf:ID="HolidayPackageService">
      <process:hasOutput rdf:resource="#HolidayCost"/>
      <process:hasInput rdf:resource="#Return"/>
      <process:hasInput rdf:resource="#Persons"/>
      <process:hasInput rdf:resource="#Origin"/>
      <process:hasInput rdf:resource="#Guestid"/>
      <process:hasOutput rdf:resource="#HotelAddress"/>
      <process:hasOutput rdf:resource="#HotelFacilities"/>
      <process:hasFiniteStateMachine
rdf:resource="HolidayPackageServiceFSM.owl#HolidayPackageServiceFSM"/>
      <process:hasInput rdf:resource="#Depart"/>
      <process:hasOutput rdf:resource="#HotelBrochure"/>
      <process:hasOutput rdf:resource="#SalesTax"/>
```

```
                <process:hasOutput rdf:resource="#HotelStars"/>
                <process:hasOutput rdf:resource="#FlightReservation"/>
                <process:hasOutput rdf:resource="#FlightNumber"/>
                <process:hasOutput rdf:resource="#HotelName"/>
                <process:hasOutput rdf:resource="#FlightItinerary"/>
                <process:hasOutput rdf:resource="#FlightClass"/>
                <process:composedOf>
                  <process:Sequence>
                    <process:components>
                      <process:ControlConstructList>
                        <objList:first rdf:resource="#FlightServicePerform"/>
                        <objList:rest>
                          <process:ControlConstructList>
                            <objList:first rdf:resource="#HotelServicePerform"/>
                            <objList:rest>
                              <process:ControlConstructList>
                                <objList:first rdf:resource="#HolidayBillServicePerform"/>
                                <objList:rest rdf:resource="generic/ObjectList.owl#nil"/>
                              </process:ControlConstructList>
                            </objList:rest>
                          </process:ControlConstructList>
                        </objList:rest>
                      </process:ControlConstructList>
                    </process:components>
                  </process:Sequence>
                </process:composedOf>
                <process:hasOutput rdf:resource="#HotelReservation"/>
                <process:hasInput rdf:resource="#Destination"/>
              </process:CompositeProcess>
            </process:process>
            <process:hasDataFrom>
              <process:InputBinding>
                <process:valueSource>
                  <process:ValueOf>
                    <process:theVar rdf:resource="#Guestid"/>
                    <process:fromProcess rdf:resource="#LoginServicePerform"/>
                  </process:ValueOf>
                </process:valueSource>
                <process:toParam rdf:resource="#Guestid"/>
              </process:InputBinding>
            </process:hasDataFrom>
          </process:Perform>
        </process:fromProcess>
      </process:ValueOf>
    </process:valueSource>
    <process:toParam rdf:resource="#SalesTax"/>
  </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
  <process:InputBinding>
    <process:valueSource>
      <process:ValueOf>
        <process:theVar rdf:resource="#HolidayCost"/>
        <process:fromProcess rdf:resource="#HolidayPackageServicePerform"/>
      </process:ValueOf>
    </process:valueSource>
    <process:toParam rdf:resource="#Charge"/>
  </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
```

```
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#TotalCost"/>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#TotalCost"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#Approved"/>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#Approved"/>
    </process:InputBinding>
  </process:hasDataFrom>
</process:Perform>
<process:CompositeProcess rdf:ID="PersonalHolidayService">
  <process:hasOutput rdf:resource="#HotelFacilities"/>
  <process:hasInput rdf:resource="#Password"/>
  <process:hasOutput rdf:resource="#TotalCost"/>
  <process:hasInput rdf:resource="#Depart"/>
  <process:hasOutput rdf:resource="#HotelReservation"/>
  <process:hasOutput rdf:resource="#Approved"/>
  <process:hasOutput rdf:resource="#HotelBrochure"/>
  <process:hasOutput rdf:resource="#HotelName"/>
  <process:hasOutput rdf:resource="#FlightItinerary"/>
  <process:hasInput rdf:resource="#Membership"/>
  <process:hasInput rdf:resource="#Username"/>
  <process:hasOutput rdf:resource="#HotelAddress"/>
  <process:hasInput rdf:resource="#Destination"/>
  <process:hasOutput rdf:resource="#HotelStars"/>
  <process:composedOf>
    <process:Sequence>
      <process:components>
        <process:ControlConstructList>
          <objList:first rdf:resource="#LoginServicePerform"/>
          <objList:rest>
            <process:ControlConstructList>
              <objList:first rdf:resource="#HolidayPackageServicePerform"/>
              <objList:rest>
                <process:ControlConstructList>
                  <objList:first rdf:resource="#PaymentServicePerform"/>
                  <objList:rest rdf:resource="generic/ObjectList.owl#nil"/>
                </process:ControlConstructList>
              </objList:rest>
            </process:ControlConstructList>
          </objList:rest>
        </process:ControlConstructList>
      </process:components>
    </process:Sequence>
  </process:composedOf>
  <process:hasOutput rdf:resource="#FlightReservation"/>
  <process:hasInput rdf:resource="#Persons"/>
  <process:hasInput rdf:resource="#Return"/>
```

```
    <process:hasOutput rdf:resource="#FlightNumber"/>
    <process:hasFiniteStateMachine
rdf:resource="PersonalHolidayServiceFSM.owl#PersonalHolidayServiceFSM"/>
    <process:hasOutput rdf:resource="#FlightClass"/>
    <process:hasInput rdf:resource="#Origin"/>
  </process:CompositeProcess>
</rdf:RDF>
```

PersonalHoliday Grounding

```
<rdf:RDF
    xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
    xmlns="http://www.daml.org/services/owl-s/1.1/PersonalHolidayGrounding.owl#"
    xmlns:policy="http://kdeg.cs.tcd.ie/Policy.owl#"
    xmlns:wsdldoc="http://www.daml.org/services/owl-s/1.1/PersonalHolidayGrounding.wsdl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:wsdl="http://www.daml.org/services/owl-s/1.1/PersonalHolidayGrounding.wsdl#"
    xmlns:thisprocess="http://www.daml.org/services/owl-s/1.1/PersonalHolidayProcess.owl#"
    xmlns:base="http://www.daml.org/services/owl-s/1.1/PersonalHolidayGrounding.owl"
    xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
    xmlns:grounding="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
  xml:base="http://www.daml.org/services/owl-s/1.1/PersonalHolidayGrounding.owl">
  <owl:Ontology rdf:about="">
    <owl:versionInfo></owl:versionInfo>
    <rdfs:comment></rdfs:comment>
    <owl:imports rdf:resource="Service.owl"/>
    <owl:imports rdf:resource="Process.owl"/>
    <owl:imports rdf:resource="Grounding.owl"/>
    <owl:imports rdf:resource="PersonalHolidayProcess.owl"/>
  </owl:Ontology>
  <grounding:WsdlGrounding rdf:ID="PersonalHolidayGrounding">
    <grounding:hasAtomicProcessGrounding>
      <grounding:WsdlAtomicProcessGrounding rdf:ID="LoginServiceGrounding">
        <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://schemas.wmlsoap.org/soap/http/</grounding:otherReference>
        <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://schemas.wmlsoap.org/wsdl/soap/</grounding:otherReference>
        <grounding:wsdlOperation>
          <grounding:WsdlOperationRef>
            <grounding:portType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
            >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#LoginService_PortType</grounding:portType>
            <grounding:operation rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
            >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#LoginService_Operation</grounding:operation>
          </grounding:WsdlOperationRef>
        </grounding:wsdlOperation>
        <grounding:wsdlDocument rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl</grounding:wsdlDocument>
        <grounding:wsdlReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:wsdlReference>
        <grounding:wsdlOutputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#LoginService_Output</grounding:wsdlOutputMessage>
        <grounding:wsdlInput>
```

```
            <grounding:WsdlInputMessageMap>
              <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Password"/>
              <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
              >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#pass</grounding:wsdlMessagePart>
            </grounding:WsdlInputMessageMap>
          </grounding:wsdlInput>
          <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:otherReference>
          <grounding:wsdlInput>
            <grounding:WsdlInputMessageMap>
              <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Membership"/>
              <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
              >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#member</grounding:wsdlMessagePart>
            </grounding:WsdlInputMessageMap>
          </grounding:wsdlInput>
          <grounding:owlsProcess rdf:resource="PersonalHolidayProcess.owl#LoginService"/>
          <grounding:wsdlOutput>
            <grounding:WsdlOutputMessageMap>
              <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Guestid"/>
              <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
              >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#user</grounding:wsdlMessagePart>
            </grounding:WsdlOutputMessageMap>
          </grounding:wsdlOutput>
          <grounding:wsdlInput>
            <grounding:WsdlInputMessageMap>
              <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Username"/>
              <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
              >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#name</grounding:wsdlMessagePart>
            </grounding:WsdlInputMessageMap>
          </grounding:wsdlInput>
          <grounding:wsdlInputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#LoginService_Input</grounding:wsdlInputMessage>
        </grounding:WsdlAtomicProcessGrounding>
      </grounding:hasAtomicProcessGrounding>
      <grounding:hasAtomicProcessGrounding>
        <grounding:WsdlAtomicProcessGrounding rdf:ID="FlightServiceGrounding">
          <grounding:wsdlOutput>
            <grounding:WsdlOutputMessageMap>
              <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Cost"/>
              <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
              >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#cost</grounding:wsdlMessagePart>
            </grounding:WsdlOutputMessageMap>
          </grounding:wsdlOutput>
          <grounding:owlsProcess rdf:resource="PersonalHolidayProcess.owl#FlightService"/>
          <grounding:wsdlOutput>
            <grounding:WsdlOutputMessageMap>
              <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#FlightNumber"/>
              <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
              >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#flightnumber</grounding:wsdlMessagePart>
            </grounding:WsdlOutputMessageMap>
          </grounding:wsdlOutput>
          <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://schemas.wmlsoap.org/soap/http/</grounding:otherReference>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Origin"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#origin</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#ArriveHome"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#returning</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
      </grounding:wsdlOutput>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Seats"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#seats</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#FlightClass"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#flightclass</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
      </grounding:wsdlOutput>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Return"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#leaving</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:wsdlDocument rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl</grounding:wsdlDocument>
      <grounding:wsdlOutputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#FlightService_Output</grounding:wsdlOutputMessage>
      <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:otherReference>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Destination"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#destination</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:wsdlOperation>
        <grounding:WsdlOperationRef>
          <grounding:portType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#FlightService_PortType</grounding:portType>
          <grounding:operation rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#FlightService_Operation</grounding:operation>
        </grounding:WsdlOperationRef>
      </grounding:wsdlOperation>
      <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#ReservationCode"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#reservation</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
      </grounding:wsdlOutput>
      <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Itinerary"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#itinerary</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
      </grounding:wsdlOutput>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Depart"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#going</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Guestid"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#user</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#ArriveDestination"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#arriving</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
      </grounding:wsdlOutput>
      <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://schemas.wmlsoap.org/wsdl/soap/</grounding:otherReference>
      <grounding:wsdlReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:wsdlReference>
      <grounding:wsdlInputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#FlightService_Input</grounding:wsdlInputMessage>
    </grounding:WsdlAtomicProcessGrounding>
  </grounding:hasAtomicProcessGrounding>
  <grounding:hasAtomicProcessGrounding>
    <grounding:WsdlAtomicProcessGrounding rdf:ID="HotelServiceGrounding">
      <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
```

```
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Brochure"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#brochure</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
      </grounding:wsdlOutput>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#CheckinDate"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#checkin</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Cost"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#cost</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
      </grounding:wsdlOutput>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Guestid"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#user</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://schemas.wmlsoap.org/wsdl/soap/</grounding:otherReference>
      <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#ReservationCode"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#reservation</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
      </grounding:wsdlOutput>
      <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:otherReference>
      <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#HotelAddress"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#address</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
      </grounding:wsdlOutput>
      <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Facilities"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#facilities</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
      </grounding:wsdlOutput>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
        <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#CheckoutDate"/>
        <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#checkout</grounding:wsdlMessagePart>
      </grounding:WsdlInputMessageMap>
    </grounding:wsdlInput>
    <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://schemas.wmlsoap.org/soap/http/</grounding:otherReference>
    <grounding:wsdlOutput>
      <grounding:WsdlOutputMessageMap>
        <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#HotelStars"/>
        <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#stars</grounding:wsdlMessagePart>
      </grounding:WsdlOutputMessageMap>
    </grounding:wsdlOutput>
    <grounding:wsdlOutputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#HotelService_Output</grounding:wsdlOutputMessage>
    <grounding:wsdlReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:wsdlReference>
    <grounding:wsdlOperation>
      <grounding:WsdlOperationRef>
        <grounding:portType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#HotelService_PortType</grounding:portType>
        <grounding:operation rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#HotelService_Operation</grounding:operation>
      </grounding:WsdlOperationRef>
    </grounding:wsdlOperation>
    <grounding:wsdlInputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#HotelService_Input</grounding:wsdlInputMessage>
    <grounding:wsdlOutput>
      <grounding:WsdlOutputMessageMap>
        <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#HotelName"/>
        <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#name</grounding:wsdlMessagePart>
      </grounding:WsdlOutputMessageMap>
    </grounding:wsdlOutput>
    <grounding:wsdlInput>
      <grounding:WsdlInputMessageMap>
        <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Rooms"/>
        <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#rooms</grounding:wsdlMessagePart>
      </grounding:WsdlInputMessageMap>
    </grounding:wsdlInput>
    <grounding:owlsProcess rdf:resource="PersonalHolidayProcess.owl#HotelService"/>
    <grounding:wsdlInput>
      <grounding:WsdlInputMessageMap>
        <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Destination"/>
        <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#location</grounding:wsdlMessagePart>
      </grounding:WsdlInputMessageMap>
    </grounding:wsdlInput>
    <grounding:wsdlDocument rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
```

```
      >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl</grounding:wsdlDocument>
    </grounding:WsdlAtomicProcessGrounding>
  </grounding:hasAtomicProcessGrounding>
  <grounding:hasAtomicProcessGrounding>
    <grounding:WsdlAtomicProcessGrounding rdf:ID="HolidayBillServiceGrounding">
      <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:otherReference>
      <grounding:wsdlReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:wsdlReference>
      <grounding:wsdlInputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#HolidayBillService_Input</grounding:wsdlInputMessage>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#HotelCost"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#hotelcost</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#HolidayCost"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#bill</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
      </grounding:wsdlOutput>
      <grounding:wsdlOutputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#HolidayBillService_Output</grounding:wsdlOutputMessage>
      <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://schemas.wmlsoap.org/soap/http/</grounding:otherReference>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#FlightCost"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#flightcost</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#SalesTax"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#tax</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
      </grounding:wsdlOutput>
      <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://schemas.wmlsoap.org/wsdl/soap/</grounding:otherReference>
      <grounding:wsdlDocument rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl</grounding:wsdlDocument>
      <grounding:wsdlOperation>
        <grounding:WsdlOperationRef>
          <grounding:portType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#HolidayBillService_PortType</grounding:portType>
```

```
          <grounding:operation rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#HolidayBillService_Operation</grounding:operation>
        </grounding:WsdlOperationRef>
       </grounding:wsdlOperation>
       <grounding:owlsProcess rdf:resource="PersonalHolidayProcess.owl#HolidayBillService"/>
     </grounding:WsdlAtomicProcessGrounding>
    </grounding:hasAtomicProcessGrounding>
    <grounding:hasAtomicProcessGrounding>
     <grounding:WsdlAtomicProcessGrounding rdf:ID="PaymentServiceGrounding">
       <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Charge"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#bill</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
       </grounding:wsdlInput>
       <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Approved"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#approved</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
       </grounding:wsdlOutput>
       <grounding:wsdlOperation>
        <grounding:WsdlOperationRef>
          <grounding:portType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#PaymentService_PortType</grounding:portType>
          <grounding:operation rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#PaymentService_Operation</grounding:operation>
        </grounding:WsdlOperationRef>
       </grounding:wsdlOperation>
       <grounding:wsdlReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
       >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:wsdlReference>
       <grounding:wsdlDocument rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
       >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl</grounding:wsdlDocument>
       <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
       >http://schemas.wmlsoap.org/soap/http/</grounding:otherReference>
       <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#Guestid"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#user</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
       </grounding:wsdlInput>
       <grounding:owlsProcess rdf:resource="PersonalHolidayProcess.owl#PaymentService"/>
       <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#SalesTax"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#tax</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
       </grounding:wsdlInput>
```

```
      <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
          <grounding:owlsParameter rdf:resource="PersonalHolidayProcess.owl#TotalCost"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#total</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
      </grounding:wsdlOutput>
      <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://schemas.wmlsoap.org/wsdl/soap/</grounding:otherReference>
      <grounding:wsdlInputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#PaymentService_Input</grounding:wsdlInputMessage>
      <grounding:wsdlOutputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.daml.org/services/owl-
s/1.1/PersonalHolidayGrounding.wsdl#PaymentService_Output</grounding:wsdlOutputMessage>
      <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:otherReference>
    </grounding:WsdlAtomicProcessGrounding>
  </grounding:hasAtomicProcessGrounding>
 </grounding:WsdlGrounding>
</rdf:RDF>
```

PersonalHoliday Service's FSM

```
<rdf:RDF
   xmlns:fsm="http://kdeg.cs.tcd.ie/FiniteStateMachine.owl#"
   xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
   xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
   xmlns:policy="http://kdeg.cs.tcd.ie/Policy.owl#"
   xmlns:grounding="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:base="http://www.daml.org/services/owl-s/1.1/PersonalHolidayServiceFSM.owl"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
   xmlns="http://www.daml.org/services/owl-s/1.1/PersonalHolidayServiceFSM.owl#"
   xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
 xml:base="http://www.daml.org/services/owl-s/1.1/PersonalHolidayServiceFSM.owl">
 <owl:Ontology rdf:about="">
   <owl:imports rdf:resource="http://kdeg.cs.tcd.ie/FiniteStateMachine.owl"/>
 </owl:Ontology>
 <fsm:Signal rdf:ID="InputEventSource">
   <fsm:value>InputSignal</fsm:value>
 </fsm:Signal>
 <fsm:SignalEvent rdf:ID="InitialEvent">
   <fsm:signal>
     <fsm:Signal rdf:ID="InitialEventSource">
       <fsm:value>InitialSignal</fsm:value>
     </fsm:Signal>
   </fsm:signal>
 </fsm:SignalEvent>
 <fsm:StateMachine rdf:ID="BusinessClass">
   <fsm:top>
     <fsm:CompositeState rdf:ID="BusinessClassCS">
       <fsm:subvertex>
         <fsm:PseudoState rdf:ID="BusinessClassInitialState">
           <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
           <fsm:outgoing>
```

```
      <fsm:Transition rdf:ID="BusinessClassInitialStateInitialTransition">
       <fsm:trigger rdf:resource="#InitialEvent"/>
       <fsm:source rdf:resource="#BusinessClassInitialState"/>
       <fsm:target>
        <fsm:FinalState rdf:ID="BusinessClassState1">
         <fsm:doActivity>BusinessClass()</fsm:doActivity>
         <fsm:doActivity>BusinessSuite()</fsm:doActivity>
         <fsm:incoming rdf:resource="#BusinessClassInitialStateInitialTransition"/>
        </fsm:FinalState>
       </fsm:target>
      </fsm:Transition>
     </fsm:outgoing>
    </fsm:PseudoState>
   </fsm:subvertex>
   <fsm:subvertex rdf:resource="#BusinessClassState1"/>
  </fsm:CompositeState>
 </fsm:top>
 <fsm:comment></fsm:comment>
</fsm:StateMachine>
<fsm:Signal rdf:ID="ProcessEventSource">
 <fsm:value>ProcessSignal</fsm:value>
</fsm:Signal>
<fsm:SignalEvent rdf:ID="OutputEvent">
 <fsm:signal>
  <fsm:Signal rdf:ID="OutputEventSource">
   <fsm:value>OutputSignal</fsm:value>
  </fsm:Signal>
 </fsm:signal>
</fsm:SignalEvent>
<fsm:CompositeState rdf:ID="HighClassCS">
 <fsm:subvertex>
  <fsm:PseudoState rdf:ID="HighClassInitialState">
   <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
   <fsm:outgoing>
    <fsm:Transition rdf:ID="HighClassInitialStateInitialTransition">
     <fsm:trigger rdf:resource="#InitialEvent"/>
     <fsm:source rdf:resource="#HighClassInitialState"/>
     <fsm:target>
      <fsm:FinalState rdf:ID="HighClassState1">
       <fsm:doActivity>FirstClass()</fsm:doActivity>
       <fsm:doActivity>PresidentialSuite()</fsm:doActivity>
       <fsm:incoming rdf:resource="#HighClassInitialStateInitialTransition"/>
      </fsm:FinalState>
     </fsm:target>
    </fsm:Transition>
   </fsm:outgoing>
  </fsm:PseudoState>
 </fsm:subvertex>
 <fsm:subvertex rdf:resource="#HighClassState1"/>
</fsm:CompositeState>
<fsm:Transition rdf:ID="AccessibilityInitialStateInitialTransition">
 <fsm:trigger rdf:resource="#InitialEvent"/>
 <fsm:source>
  <fsm:PseudoState rdf:ID="AccessibilityInitialState">
   <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
   <fsm:outgoing rdf:resource="#AccessibilityInitialStateInitialTransition"/>
  </fsm:PseudoState>
 </fsm:source>
 <fsm:target>
  <fsm:FinalState rdf:ID="AccessibilityState1">
```

```
      <fsm:doActivity>Accessibility()</fsm:doActivity>
      <fsm:incoming rdf:resource="#AccessibilityInitialStateInitialTransition"/>
    </fsm:FinalState>
  </fsm:target>
</fsm:Transition>
<fsm:Transition rdf:ID="MasterCardInitialStateInitialTransition">
  <fsm:trigger rdf:resource="#InitialEvent"/>
  <fsm:source>
    <fsm:PseudoState rdf:ID="MasterCardInitialState">
      <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
      <fsm:outgoing rdf:resource="#MasterCardInitialStateInitialTransition"/>
    </fsm:PseudoState>
  </fsm:source>
  <fsm:target>
    <fsm:FinalState rdf:ID="MasterCardState1">
      <fsm:doActivity>paymentMethod(Master)</fsm:doActivity>
      <fsm:incoming rdf:resource="#MasterCardInitialStateInitialTransition"/>
    </fsm:FinalState>
  </fsm:target>
</fsm:Transition>
<fsm:FinalState rdf:ID="VegetarianState1">
  <fsm:doActivity>VegetarianMeal()</fsm:doActivity>
  <fsm:doActivity>VegetarianMenu()</fsm:doActivity>
  <fsm:incoming>
    <fsm:Transition rdf:ID="VegetarianInitialStateInitialTransition">
      <fsm:trigger rdf:resource="#InitialEvent"/>
      <fsm:source>
        <fsm:PseudoState rdf:ID="VegetarianInitialState">
          <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
          <fsm:outgoing rdf:resource="#VegetarianInitialStateInitialTransition"/>
        </fsm:PseudoState>
      </fsm:source>
      <fsm:target rdf:resource="#VegetarianState1"/>
    </fsm:Transition>
  </fsm:incoming>
</fsm:FinalState>
<fsm:CompositeState rdf:ID="ServiceState">
  <fsm:subvertex>
    <fsm:PseudoState rdf:ID="InitialState">
      <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
      <fsm:outgoing>
        <fsm:Transition rdf:ID="InitialT">
          <fsm:trigger rdf:resource="#InitialEvent"/>
          <fsm:source rdf:resource="#InitialState"/>
          <fsm:target>
            <fsm:CompositeState rdf:ID="IdleState">
              <fsm:incoming rdf:resource="#InitialT"/>
              <fsm:outgoing>
                <fsm:Transition rdf:ID="IdleToInput">
                  <fsm:trigger>
                    <fsm:SignalEvent rdf:ID="InputEvent">
                      <fsm:signal rdf:resource="#InputEventSource"/>
                    </fsm:SignalEvent>
                  </fsm:trigger>
                  <fsm:source rdf:resource="#IdleState"/>
                  <fsm:target>
                    <fsm:CompositeState rdf:ID="InputState">
                      <fsm:incoming rdf:resource="#IdleToInput"/>
                      <fsm:outgoing>
                        <fsm:Transition rdf:ID="InputToProcess">
```

```
<fsm:trigger>
 <fsm:SignalEvent rdf:ID="ProcessEvent">
  <fsm:signal rdf:resource="#ProcessEventSource"/>
 </fsm:SignalEvent>
</fsm:trigger>
<fsm:source rdf:resource="#InputState"/>
<fsm:target>
 <fsm:CompositeState rdf:ID="ProcessState">
  <fsm:outgoing>
   <fsm:Transition rdf:ID="ProcessToOutput">
    <fsm:trigger rdf:resource="#OutputEvent"/>
    <fsm:source rdf:resource="#ProcessState"/>
    <fsm:target>
     <fsm:CompositeState rdf:ID="OutputState">
      <fsm:incoming rdf:resource="#ProcessToOutput"/>
      <fsm:outgoing>
       <fsm:Transition rdf:ID="OutputToIdle">
        <fsm:trigger>
         <fsm:SignalEvent rdf:ID="IdleEvent">
          <fsm:signal>
           <fsm:Signal rdf:ID="IdleEventSource">
            <fsm:value
            >IdleSignal</fsm:value>
           </fsm:Signal>
          </fsm:signal>
         </fsm:SignalEvent>
        </fsm:trigger>
        <fsm:source rdf:resource="#OutputState"/>
        <fsm:target rdf:resource="#IdleState"/>
       </fsm:Transition>
      </fsm:outgoing>
     </fsm:CompositeState>
    </fsm:target>
   </fsm:Transition>
  </fsm:outgoing>
  <fsm:subvertex>
   <fsm:SubmachineState rdf:ID="TravelVoucherSubSM">
    <fsm:submachine>
     <fsm:StateMachine rdf:ID="TravelVoucher">
      <fsm:top>
       <fsm:CompositeState rdf:ID="TravelVoucherCS">
        <fsm:subvertex>
         <fsm:PseudoState rdf:ID="TravelVoucherInitialState">
          <fsm:pseudoStateKind
          >initial</fsm:pseudoStateKind>
          <fsm:outgoing>
           <fsm:Transition rdf:ID="TravelVoucherInitialStateInitialTransition">
            <fsm:trigger rdf:resource="#InitialEvent"/>
            <fsm:source rdf:resource="#TravelVoucherInitialState"/>
            <fsm:target>
             <fsm:FinalState rdf:ID="TravelVoucherState1">
              <fsm:doActivity
              >paymentMethod(Voucher)</fsm:doActivity>
              <fsm:incoming
rdf:resource="#TravelVoucherInitialStateInitialTransition"/>
             </fsm:FinalState>
            </fsm:target>
           </fsm:Transition>
          </fsm:outgoing>
         </fsm:PseudoState>
      </fsm:subvertex>
      <fsm:subvertex rdf:resource="#TravelVoucherState1"/>
     </fsm:CompositeState>
    </fsm:top>
    <fsm:comment>Pay using voucher</fsm:comment>
   </fsm:StateMachine>
  </fsm:submachine>
 </fsm:SubmachineState>
</fsm:subvertex>
<fsm:subvertex>
 <fsm:SubmachineState rdf:ID="VegetarianSubSM">
  <fsm:submachine>
   <fsm:StateMachine rdf:ID="Vegetarian">
    <fsm:top>
     <fsm:CompositeState rdf:ID="VegetarianCS">
      <fsm:subvertex rdf:resource="#VegetarianInitialState"/>
      <fsm:subvertex rdf:resource="#VegetarianState1"/>
     </fsm:CompositeState>
    </fsm:top>
    <fsm:comment></fsm:comment>
   </fsm:StateMachine>
  </fsm:submachine>
 </fsm:SubmachineState>
</fsm:subvertex>
<fsm:subvertex>
 <fsm:SubmachineState rdf:ID="BusinessClassSubSM">
  <fsm:submachine rdf:resource="#BusinessClass"/>
 </fsm:SubmachineState>
</fsm:subvertex>
<fsm:subvertex>
 <fsm:SubmachineState rdf:ID="MasterCardSubSM">
  <fsm:submachine>
   <fsm:StateMachine rdf:ID="MasterCard">
    <fsm:top>
     <fsm:CompositeState rdf:ID="MasterCardCS">
      <fsm:subvertex rdf:resource="#MasterCardInitialState"/>
      <fsm:subvertex rdf:resource="#MasterCardState1"/>
     </fsm:CompositeState>
    </fsm:top>
    <fsm:comment>Pay using MasterCard</fsm:comment>
   </fsm:StateMachine>
  </fsm:submachine>
 </fsm:SubmachineState>
</fsm:subvertex>
<fsm:subvertex>
 <fsm:SubmachineState rdf:ID="AmericanExpressSubSM">
  <fsm:submachine>
   <fsm:StateMachine rdf:ID="AmericanExpress">
    <fsm:top>
     <fsm:CompositeState rdf:ID="AmericanExpressCS">
      <fsm:subvertex>
       <fsm:PseudoState rdf:ID="AmericanExpressInitialState">
        <fsm:pseudoStateKind
        >initial</fsm:pseudoStateKind>
        <fsm:outgoing>
         <fsm:Transition
rdf:ID="AmericanExpressInitialStateInitialTransition">
          <fsm:trigger rdf:resource="#InitialEvent"/>
          <fsm:source rdf:resource="#AmericanExpressInitialState"/>
          <fsm:target>
```

```
                          <fsm:FinalState rdf:ID="AmericanExpressState1">
                           <fsm:doActivity
                           >paymentMethod(American)</fsm:doActivity>
                           <fsm:incoming
rdf:resource="#AmericanExpressInitialStateInitialTransition"/>
                          </fsm:FinalState>
                         </fsm:target>
                        </fsm:Transition>
                       </fsm:outgoing>
                      </fsm:PseudoState>
                     </fsm:subvertex>
                     <fsm:subvertex rdf:resource="#AmericanExpressState1"/>
                    </fsm:CompositeState>
                   </fsm:top>
                   <fsm:comment>Pay using American Express</fsm:comment>
                  </fsm:StateMachine>
                 </fsm:submachine>
                </fsm:SubmachineState>
               </fsm:subvertex>
               <fsm:subvertex>
                <fsm:SubmachineState rdf:ID="AccessibilitySubSM">
                 <fsm:submachine>
                  <fsm:StateMachine rdf:ID="Accessibility">
                   <fsm:top>
                    <fsm:CompositeState rdf:ID="AccessibilityCS">
                     <fsm:subvertex rdf:resource="#AccessibilityInitialState"/>
                     <fsm:subvertex rdf:resource="#AccessibilityState1"/>
                    </fsm:CompositeState>
                   </fsm:top>
                   <fsm:comment></fsm:comment>
                  </fsm:StateMachine>
                 </fsm:submachine>
                </fsm:SubmachineState>
               </fsm:subvertex>
               <fsm:subvertex>
                <fsm:SubmachineState rdf:ID="HighClassSubSM">
                 <fsm:submachine>
                  <fsm:StateMachine rdf:ID="HighClass">
                   <fsm:top rdf:resource="#HighClassCS"/>
                   <fsm:comment></fsm:comment>
                   <fsm:comment>High class room - presidential suite</fsm:comment>
                  </fsm:StateMachine>
                 </fsm:submachine>
                </fsm:SubmachineState>
               </fsm:subvertex>
               <fsm:incoming rdf:resource="#InputToProcess"/>
              </fsm:CompositeState>
             </fsm:target>
            </fsm:Transition>
           </fsm:outgoing>
          </fsm:CompositeState>
         </fsm:target>
        </fsm:Transition>
       </fsm:outgoing>
       <fsm:incoming rdf:resource="#OutputToIdle"/>
      </fsm:CompositeState>
     </fsm:target>
    </fsm:Transition>
   </fsm:outgoing>
  </fsm:PseudoState>
```

```
    </fsm:subvertex>
    <fsm:subvertex rdf:resource="#IdleState"/>
    <fsm:subvertex rdf:resource="#ProcessState"/>
    <fsm:subvertex rdf:resource="#InputState"/>
    <fsm:subvertex rdf:resource="#OutputState"/>
  </fsm:CompositeState>
  <fsm:StateMachine rdf:ID="PersonalHolidayServiceFSM">
    <fsm:top rdf:resource="#ServiceState"/>
  </fsm:StateMachine>
</rdf:RDF>
```

PersonalHoliday Service's Management Policy

```
<rdf:RDF
    xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
    xmlns:policy="http://kdeg.cs.tcd.ie/Policy.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:thisprocess="http://www.daml.org/services/owl-s/1.1/PersonalHolidayProcess.owl#"
    xmlns="http://www.daml.org/services/owl-s/1.1/PersonalHolidayServicePolicy.owl#"
    xmlns:fsm="http://kdeg.cs.tcd.ie/FiniteStateMachine.owl#"
    xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
    xmlns:grounding="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:base="http://www.daml.org/services/owl-s/1.1/PersonalHolidayServicePolicy.owl"
    xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
  xml:base="http://www.daml.org/services/owl-s/1.1/PersonalHolidayServicePolicy.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://kdeg.cs.tcd.ie/Policy.owl"/>
    <owl:imports rdf:resource="PersonalHolidayProcess.owl"/>
  </owl:Ontology>
  <policy:SimpleCondition rdf:ID="PersonalHolidayServicePolicy1Condition102">
    <policy:subject>
      <policy:Subject rdf:ID="Membership"/>
    </policy:subject>
    <policy:predicate>
      <policy:Predicate rdf:ID="equal"/>
    </policy:predicate>
    <policy:value>FrequentFlyer</policy:value>
  </policy:SimpleCondition>
  <policy:SimpleEvent rdf:ID="PersonalHolidayServicePolicy2Event1">
    <policy:value>ProcessEvent</policy:value>
  </policy:SimpleEvent>
  <policy:ComplexCondition rdf:ID="PersonalHolidayServicePolicy1Condition1">
    <rdfs:first>
      <policy:AndList>
        <rdfs:first>
          <policy:SimpleCondition rdf:ID="PersonalHolidayServicePolicy1Condition101">
            <policy:subject>
              <policy:Subject rdf:ID="Username"/>
            </policy:subject>
            <policy:predicate rdf:resource="#equal"/>
            <policy:value>Susan.Smith</policy:value>
          </policy:SimpleCondition>
        </rdfs:first>
        <rdfs:rest rdf:resource="#PersonalHolidayServicePolicy1Condition102"/>
      </policy:AndList>
    </rdfs:first>
    <rdfs:rest></rdfs:rest>
  </policy:ComplexCondition>
  <policy:SimpleCondition rdf:ID="PersonalHolidayServicePolicy2Condition1">
    <policy:subject rdf:resource="#Membership"/>
    <policy:predicate rdf:resource="#equal"/>
    <policy:value>SalesGroup</policy:value>
  </policy:SimpleCondition>
  <policy:Policy rdf:ID="PersonalHolidayServicePolicy1">
    <rdfs:comment></rdfs:comment>
    <policy:target rdf:resource="PersonalHolidayProcess.owl#PersonalHolidayService"/>
    <policy:event>
      <policy:SimpleEvent rdf:ID="PersonalHolidayServicePolicy1Event1">
        <policy:value>ProcessEvent</policy:value>
      </policy:SimpleEvent>
    </policy:event>
    <policy:condition rdf:resource="#PersonalHolidayServicePolicy1Condition1"/>
    <policy:action>
      <policy:SimpleAction rdf:ID="PersonalHolidayServicePolicy1Action1">
        <policy:value>Accessibility</policy:value>
      </policy:SimpleAction>
    </policy:action>
  </policy:Policy>
  <policy:SimpleAction rdf:ID="PersonalHolidayServicePolicy2Action1">
    <policy:value>BusinessClass</policy:value>
  </policy:SimpleAction>
  <policy:Policy rdf:ID="PersonalHolidayServicePolicy2">
    <rdfs:comment></rdfs:comment>
    <policy:target rdf:resource="PersonalHolidayProcess.owl#PersonalHolidayService"/>
    <policy:event rdf:resource="#PersonalHolidayServicePolicy2Event1"/>
    <policy:condition rdf:resource="#PersonalHolidayServicePolicy2Condition1"/>
    <policy:action rdf:resource="#PersonalHolidayServicePolicy2Action1"/>
  </policy:Policy>
</rdf:RDF>
```

PersonalHoliday Service's Refined Policies

```
<rdf:RDF
   xmlns="http://www.daml.org/services/owl-s/1.1/PersonalHolidayServiceRefinedPolicy.owl#"
   xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
   xmlns:policy="http://kdeg.cs.tcd.ie/Policy.owl#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:thisprocess="http://www.daml.org/services/owl-s/1.1/PersonalHolidayProcess.owl#"
   xmlns:fsm="http://kdeg.cs.tcd.ie/FiniteStateMachine.owl#"
   xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
   xmlns:grounding="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
   xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
   xmlns:base="http://www.daml.org/services/owl-s/1.1/PersonalHolidayServiceRefinedPolicy.owl"
  xml:base="http://www.daml.org/services/owl-s/1.1/PersonalHolidayServiceRefinedPolicy.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://kdeg.cs.tcd.ie/Policy.owl"/>
    <owl:imports rdf:resource="PersonalHolidayProcess.owl"/>
  </owl:Ontology>
  <policy:SimpleAction rdf:ID="PersonalHolidayServicePolicy1Action102">
    <policy:value>event(VegetarianEvent)</policy:value>
  </policy:SimpleAction>
  <policy:Policy rdf:ID="LoginServicePolicy1">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="PersonalHolidayProcess.owl#LoginService"/>
  <policy:event>
   <policy:SimpleEvent rdf:ID="LoginServicePolicy1Event1C">
    <policy:value>RPersonalHolidayServicePolicy1Event1C</policy:value>
   </policy:SimpleEvent>
  </policy:event>
  <policy:condition>
   <policy:SimpleCondition rdf:ID="LoginServicePolicy1Condition1">
    <policy:subject>
     <policy:Subject rdf:ID="name"/>
    </policy:subject>
    <policy:predicate>
     <policy:Predicate rdf:ID="equal"/>
    </policy:predicate>
    <policy:value>John.Murphy</policy:value>
   </policy:SimpleCondition>
  </policy:condition>
  <policy:action>
   <policy:SimpleAction rdf:ID="LoginServicePolicy1Action1CU0">
    <policy:value>event(RPersonalHolidayServicePolicy1Event1CU0)</policy:value>
   </policy:SimpleAction>
  </policy:action>
 </policy:Policy>
 <policy:SimpleAction rdf:ID="HolidayPackageServicePolicy1CAction1">
  <policy:value>event(HolidayPackageServicePolicy1CEvent2A)</policy:value>
 </policy:SimpleAction>
 <policy:Policy rdf:ID="HolidayPackageServicePolicy2C">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="PersonalHolidayProcess.owl#HolidayPackageService"/>
  <policy:event>
   <policy:SimpleEvent rdf:ID="HolidayPackageServicePolicy2CEvent1A">
    <policy:value>RPersonalHolidayServicePolicy2Event1A</policy:value>
   </policy:SimpleEvent>
  </policy:event>
  <policy:action>
   <policy:SimpleAction rdf:ID="HolidayPackageServicePolicy2CAction1">
    <policy:value>event(HolidayPackageServicePolicy2CEvent2A)</policy:value>
   </policy:SimpleAction>
  </policy:action>
 </policy:Policy>
 <policy:SimpleAction rdf:ID="VegetarianAction1">
  <policy:value>VegetarianMeal()</policy:value>
  <policy:value>VegetarianMenu()</policy:value>
 </policy:SimpleAction>
 <policy:SimpleAction rdf:ID="VegetarianEvent1">
  <policy:value>VegetarianEvent</policy:value>
 </policy:SimpleAction>
 <policy:Policy rdf:ID="RPersonalHolidayServicePolicy2">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="PersonalHolidayProcess.owl#PersonalHolidayService"/>
  <policy:event>
   <policy:SimpleEvent rdf:ID="RPersonalHolidayServicePolicy2Event1">
    <policy:value>ProcessEvent</policy:value>
   </policy:SimpleEvent>
  </policy:event>
  <policy:action>
   <policy:SimpleAction rdf:ID="RPersonalHolidayServicePolicy2Action1C">
    <policy:value>event(RPersonalHolidayServicePolicy2Event1C)</policy:value>
   </policy:SimpleAction>
```

```
    </policy:action>
  </policy:Policy>
  <policy:Policy rdf:ID="FlightServicePolicy2CC">
    <rdfs:comment></rdfs:comment>
    <policy:target rdf:resource="PersonalHolidayProcess.owl#FlightService"/>
    <policy:event>
      <policy:SimpleEvent rdf:ID="FlightServicePolicy2CCEvent2A">
        <policy:value>HolidayPackageServicePolicy2CEvent2A</policy:value>
      </policy:SimpleEvent>
    </policy:event>
    <policy:action>
      <policy:SimpleAction rdf:ID="FlightServicePolicy2CCAction2">
        <policy:value>event(BusinessClassEvent)</policy:value>
      </policy:SimpleAction>
    </policy:action>
  </policy:Policy>
  <policy:SimpleAction rdf:ID="PersonalHolidayServicePolicy1Action101">
    <policy:value>event(AccessibilityEvent)</policy:value>
  </policy:SimpleAction>
  <policy:ComplexEvent rdf:ID="BusinessClassEvent1a">
    <rdfs:first>
      <policy:AndList>
        <rdfs:first>
          <policy:SimpleEvent rdf:ID="BusinessClassEvent1b">
            <policy:value>ProcessEvent</policy:value>
          </policy:SimpleEvent>
        </rdfs:first>
        <rdfs:rest>
          <policy:SimpleAction rdf:ID="BusinessClassEvent1">
            <policy:value>BusinessClassEvent</policy:value>
          </policy:SimpleAction>
        </rdfs:rest>
      </policy:AndList>
    </rdfs:first>
    <rdfs:rest></rdfs:rest>
    <rdfs:first>
      <policy:AndList>
        <rdfs:first rdf:resource="#BusinessClassEvent1b"/>
        <rdfs:rest rdf:resource="#BusinessClassEvent1"/>
      </policy:AndList>
    </rdfs:first>
  </policy:ComplexEvent>
  <policy:SimpleAction rdf:ID="RPersonalHolidayServicePolicy2Action1CU">
    <policy:value>event(RPersonalHolidayServicePolicy2Event1A)</policy:value>
  </policy:SimpleAction>
  <policy:Policy rdf:ID="RPersonalHolidayServicePolicy2CU">
    <rdfs:comment></rdfs:comment>
    <policy:target rdf:resource="PersonalHolidayProcess.owl#PersonalHolidayService"/>
    <policy:event>
      <policy:SimpleEvent rdf:ID="LoginServicePolicy2Event1CU">
        <policy:value>RPersonalHolidayServicePolicy2Event1CU0</policy:value>
      </policy:SimpleEvent>
    </policy:event>
    <policy:action rdf:resource="#RPersonalHolidayServicePolicy2Action1CU"/>
  </policy:Policy>
  <policy:Policy rdf:ID="HotelServicePolicy2CC">
    <rdfs:comment></rdfs:comment>
    <policy:target rdf:resource="PersonalHolidayProcess.owl#HotelService"/>
    <policy:event>
      <policy:SimpleEvent rdf:ID="HotelServicePolicy2CCEvent2A">
      <policy:value>HolidayPackageServicePolicy2CEvent2A</policy:value>
    </policy:SimpleEvent>
  </policy:event>
  <policy:action>
    <policy:SimpleAction rdf:ID="HotelServicePolicy2CCAction2">
      <policy:value>event(BusinessClassEvent)</policy:value>
    </policy:SimpleAction>
  </policy:action>
</policy:Policy>
<policy:Policy rdf:ID="RPersonalHolidayServicePolicy1CU">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="PersonalHolidayProcess.owl#PersonalHolidayService"/>
  <policy:event>
    <policy:SimpleEvent rdf:ID="LoginServicePolicy1Event1CU">
      <policy:value>RPersonalHolidayServicePolicy1Event1CU0</policy:value>
    </policy:SimpleEvent>
  </policy:event>
  <policy:action>
    <policy:SimpleAction rdf:ID="RPersonalHolidayServicePolicy1Action1CU">
      <policy:value>event(RPersonalHolidayServicePolicy1Event1A)</policy:value>
    </policy:SimpleAction>
  </policy:action>
</policy:Policy>
<policy:SimpleEvent rdf:ID="LoginServicePolicy2Event1C">
  <policy:value>RPersonalHolidayServicePolicy2Event1C</policy:value>
</policy:SimpleEvent>
<policy:ComplexEvent rdf:ID="VegetarianEvent1a">
  <rdfs:first>
    <policy:AndList>
      <rdfs:first>
        <policy:SimpleEvent rdf:ID="VegetarianEvent1b">
          <policy:value>ProcessEvent</policy:value>
        </policy:SimpleEvent>
      </rdfs:first>
      <rdfs:rest rdf:resource="#VegetarianEvent1"/>
    </policy:AndList>
  </rdfs:first>
  <rdfs:rest></rdfs:rest>
  <rdfs:first>
    <policy:AndList>
      <rdfs:first rdf:resource="#VegetarianEvent1b"/>
      <rdfs:rest rdf:resource="#VegetarianEvent1"/>
    </policy:AndList>
  </rdfs:first>
</policy:ComplexEvent>
<policy:SimpleAction rdf:ID="LoginServicePolicy2Action1CU0">
  <policy:value>event(RPersonalHolidayServicePolicy2Event1CU0)</policy:value>
</policy:SimpleAction>
<policy:ComplexAction rdf:ID="FlightServicePolicy1CCAction2">
  <rdfs:first>
    <policy:AndList>
      <rdfs:first rdf:resource="#PersonalHolidayServicePolicy1Action101"/>
      <rdfs:rest rdf:resource="#PersonalHolidayServicePolicy1Action102"/>
    </policy:AndList>
  </rdfs:first>
  <rdfs:rest></rdfs:rest>
</policy:ComplexAction>
<policy:Policy rdf:ID="AccessibilityPolicy1">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="PersonalHolidayProcess.owl#FlightService"/>
  <policy:event>
   <policy:ComplexEvent rdf:ID="AccessibilityEvent1a">
    <rdfs:first>
     <policy:AndList>
      <rdfs:first>
       <policy:SimpleEvent rdf:ID="AccessibilityEvent1b">
        <policy:value>ProcessEvent</policy:value>
       </policy:SimpleEvent>
      </rdfs:first>
      <rdfs:rest>
       <policy:SimpleAction rdf:ID="AccessibilityEvent1">
        <policy:value>AccessibilityEvent</policy:value>
       </policy:SimpleAction>
      </rdfs:rest>
     </policy:AndList>
    </rdfs:first>
    <rdfs:rest></rdfs:rest>
    <rdfs:first>
     <policy:AndList>
      <rdfs:first rdf:resource="#AccessibilityEvent1b"/>
      <rdfs:rest rdf:resource="#AccessibilityEvent1"/>
     </policy:AndList>
    </rdfs:first>
   </policy:ComplexEvent>
  </policy:event>
  <policy:action>
   <policy:SimpleAction rdf:ID="AccessibilityAction1">
    <policy:value>Accessibility()</policy:value>
   </policy:SimpleAction>
  </policy:action>
  <policy:target rdf:resource="PersonalHolidayProcess.owl#HotelService"/>
 </policy:Policy>
 <policy:SimpleEvent rdf:ID="FlightServicePolicy1CCEvent2A">
  <policy:value>HolidayPackageServicePolicy1CEvent2A</policy:value>
 </policy:SimpleEvent>
 <policy:SimpleEvent rdf:ID="RPersonalHolidayServicePolicy1Event1">
  <policy:value>ProcessEvent</policy:value>
 </policy:SimpleEvent>
 <policy:SimpleCondition rdf:ID="PersonalHolidayServicePolicy2Condition102">
  <policy:subject>
   <policy:Subject rdf:ID="member"/>
  </policy:subject>
  <policy:predicate rdf:resource="#equal"/>
  <policy:value>FrequentFlyer</policy:value>
 </policy:SimpleCondition>
 <policy:Policy rdf:ID="FlightServicePolicy1CC">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="PersonalHolidayProcess.owl#FlightService"/>
  <policy:event rdf:resource="#FlightServicePolicy1CCEvent2A"/>
  <policy:action rdf:resource="#FlightServicePolicy1CCAction2"/>
 </policy:Policy>
 <policy:Policy rdf:ID="RPersonalHolidayServicePolicy1">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="PersonalHolidayProcess.owl#PersonalHolidayService"/>
  <policy:event rdf:resource="#RPersonalHolidayServicePolicy1Event1"/>
  <policy:action>
   <policy:SimpleAction rdf:ID="RPersonalHolidayServicePolicy1Action1C">
    <policy:value>event(RPersonalHolidayServicePolicy1Event1C)</policy:value>
   </policy:SimpleAction>
  </policy:action>
```

```
</policy:Policy>
<policy:ComplexAction rdf:ID="HotelServicePolicy1CCAction2">
 <rdfs:first>
   <policy:AndList>
     <rdfs:first rdf:resource="#PersonalHolidayServicePolicy1Action101"/>
     <rdfs:rest rdf:resource="#PersonalHolidayServicePolicy1Action102"/>
   </policy:AndList>
 </rdfs:first>
 <rdfs:rest></rdfs:rest>
</policy:ComplexAction>
<policy:SimpleAction rdf:ID="BusinessClassAction1">
 <policy:value>BusinessClass()</policy:value>
 <policy:value>BusinessSuite()</policy:value>
</policy:SimpleAction>
<policy:Policy rdf:ID="HotelServicePolicy1CC">
 <rdfs:comment></rdfs:comment>
 <policy:target rdf:resource="PersonalHolidayProcess.owl#HotelService"/>
 <policy:event>
   <policy:SimpleEvent rdf:ID="HotelServicePolicy1CCEvent2A">
     <policy:value>HolidayPackageServicePolicy1CEvent2A</policy:value>
   </policy:SimpleEvent>
 </policy:event>
 <policy:action rdf:resource="#HotelServicePolicy1CCAction2"/>
</policy:Policy>
<policy:Policy rdf:ID="BusinessClassPolicy1">
 <rdfs:comment></rdfs:comment>
 <policy:target rdf:resource="PersonalHolidayProcess.owl#FlightService"/>
 <policy:event rdf:resource="#BusinessClassEvent1a"/>
 <policy:action rdf:resource="#BusinessClassAction1"/>
 <policy:target rdf:resource="PersonalHolidayProcess.owl#HotelService"/>
</policy:Policy>
<policy:ComplexCondition rdf:ID="LoginServicePolicy2Condition1">
 <rdfs:first>
   <policy:AndList>
     <rdfs:first>
       <policy:SimpleCondition rdf:ID="PersonalHolidayServicePolicy2Condition101">
         <policy:subject rdf:resource="#name"/>
         <policy:predicate rdf:resource="#equal"/>
         <policy:value>Susan.Smith</policy:value>
       </policy:SimpleCondition>
     </rdfs:first>
     <rdfs:rest rdf:resource="#PersonalHolidayServicePolicy2Condition102"/>
   </policy:AndList>
 </rdfs:first>
 <rdfs:rest></rdfs:rest>
</policy:ComplexCondition>
<policy:Policy rdf:ID="HolidayPackageServicePolicy1C">
 <rdfs:comment></rdfs:comment>
 <policy:target rdf:resource="PersonalHolidayProcess.owl#HolidayPackageService"/>
 <policy:event>
   <policy:SimpleEvent rdf:ID="HolidayPackageServicePolicy1CEvent1A">
     <policy:value>RPersonalHolidayServicePolicy1Event1A</policy:value>
   </policy:SimpleEvent>
 </policy:event>
 <policy:action rdf:resource="#HolidayPackageServicePolicy1CAction1"/>
</policy:Policy>
<policy:Policy rdf:ID="LoginServicePolicy2">
 <rdfs:comment></rdfs:comment>
 <policy:target rdf:resource="PersonalHolidayProcess.owl#LoginService"/>
 <policy:event rdf:resource="#LoginServicePolicy2Event1C"/>
```

```
    <policy:condition rdf:resource="#LoginServicePolicy2Condition1"/>
    <policy:action rdf:resource="#LoginServicePolicy2Action1CU0"/>
  </policy:Policy>
  <policy:Policy rdf:ID="VegetarianPolicy1">
    <rdfs:comment></rdfs:comment>
    <policy:target rdf:resource="PersonalHolidayProcess.owl#FlightService"/>
    <policy:event rdf:resource="#VegetarianEvent1a"/>
    <policy:action rdf:resource="#VegetarianAction1"/>
    <policy:target rdf:resource="PersonalHolidayProcess.owl#HotelService"/>
  </policy:Policy>
</rdf:RDF>
```

PersonalHoliday Service's Refined Policies as Jess Rules

```
(defrule HotelServicePolicy2CC
(event (service ?service0&HotelService|Any) (type ?type0) (name
?event0&HolidayPackageServicePolicy2CEvent2A))
=>(assert (event (service Any) (type Policy) (name BusinessClassEvent))) (assert (action (service
?service0) (name ?event0))))
(defrule LoginServicePolicy2
(event (service ?service0&LoginService|Any) (type ?type0) (name
?event0&RPersonalHolidayServicePolicy2Event1C)) (event (service ?cservice0) (type ?ctype0) (name
?cevent0&~IdleEvent))
(and (param (service ?cservice0&LoginService) (direction ?direction0) (name ?param0&name) (value
?value0)) (test (eq ?value0 Susan.Smith)) (param (service ?cservice1&LoginService) (direction
?direction1) (name ?param1&member) (value ?value1)) (test (eq ?value1 Frequent Flyer)))
 =>(assert (event (service Any) (type Policy) (name RPersonalHolidayServicePolicy2Event1CU0)))
(assert (action (service ?service0) (name ?event0))))
(defrule RPersonalHolidayServicePolicy1
(event (service ?service0&PersonalHolidayService|Any) (type ?type0) (name ?event0&ProcessEvent))
=>(assert (event (service Any) (type Policy) (name RPersonalHolidayServicePolicy1Event1C))))
(defrule RPersonalHolidayServicePolicy2CU
(event (service ?service0&PersonalHolidayService|Any) (type ?type0) (name
?event0&RPersonalHolidayServicePolicy2Event1CU0))
=>(assert (event (service Any) (type Policy) (name RPersonalHolidayServicePolicy2Event1A))) (assert
(action (service ?service0) (name ?event0))))
(defrule BusinessClassPolicy1
(and (event (service ?service0&FlightService|Any) (type ?type0) (name ?event0&ProcessEvent))
(event (service ?service1&FlightService|Any) (type ?type1) (name ?event1&BusinessClassEvent)))
=>(service-state FlightService "BusinessClass()") (assert (action (service ?service1) (name ?event1))))
(defrule HolidayPackageServicePolicy2C
(event (service ?service0&HolidayPackageService|Any) (type ?type0) (name
?event0&RPersonalHolidayServicePolicy2Event1A))
=>(assert (event (service Any) (type Policy) (name HolidayPackageServicePolicy2CEvent2A))) (assert
(action (service ?service0) (name ?event0))))
(defrule HolidayPackageServicePolicy1C
(event (service ?service0&HolidayPackageService|Any) (type ?type0) (name
?event0&RPersonalHolidayServicePolicy1Event1A))
=>(assert (event (service Any) (type Policy) (name HolidayPackageServicePolicy1CEvent2A))) (assert
(action (service ?service0) (name ?event0))))
(defrule LoginServicePolicy1
(event (service ?service0&LoginService|Any) (type ?type0) (name
?event0&RPersonalHolidayServicePolicy1Event1C)) (event (service ?cservice0) (type ?ctype0) (name
?cevent0&~IdleEvent))
(param (service ?cservice0&LoginService) (direction ?direction0) (name ?param0&name) (value
?value0)) (test (eq ?value0 John.Murphy))
=>(assert (event (service Any) (type Policy) (name RPersonalHolidayServicePolicy1Event1CU0)))
(assert (action (service ?service0) (name ?event0))))
(defrule VegetarianPolicy1
(and (event (service ?service0&FlightService|Any) (type ?type0) (name ?event0&ProcessEvent))
(event (service ?service1&FlightService|Any) (type ?type1) (name ?event1&VegetarianEvent)))
=>(service-state FlightService "VegetarianMeal()") (assert (action (service ?service1) (name
?event1))))
(defrule RPersonalHolidayServicePolicy1CU
(event (service ?service0&PersonalHolidayService|Any) (type ?type0) (name
?event0&RPersonalHolidayServicePolicy1Event1CU0))
=>(assert (event (service Any) (type Policy) (name RPersonalHolidayServicePolicy1Event1A))) (assert
(action (service ?service0) (name ?event0))))
(defrule FlightServicePolicy2CC
(event (service ?service0&FlightService|Any) (type ?type0) (name
?event0&HolidayPackageServicePolicy2CEvent2A))
=>(assert (event (service Any) (type Policy) (name BusinessClassEvent))) (assert (action (service
?service0) (name ?event0))))
(defrule FlightServicePolicy1CC
(event (service ?service0&FlightService|Any) (type ?type0) (name
?event0&HolidayPackageServicePolicy1CEvent2A))
=>(and (assert (event (service Any) (type Policy) (name AccessibilityEvent))) (assert (event (service
Any) (type Policy) (name VegetarianEvent)))) (assert (action (service ?service0) (name ?event0))))
(defrule HotelServicePolicy1CC
(event (service ?service0&HotelService|Any) (type ?type0) (name
?event0&HolidayPackageServicePolicy1CEvent2A))
=>(and (assert (event (service Any) (type Policy) (name AccessibilityEvent))) (assert (event (service
Any) (type Policy) (name VegetarianEvent)))) (assert (action (service ?service0) (name ?event0))))
(defrule RPersonalHolidayServicePolicy2
(event (service ?service0&PersonalHolidayService|Any) (type ?type0) (name ?event0&ProcessEvent))
=>(assert (event (service Any) (type Policy) (name RPersonalHolidayServicePolicy2Event1C))))
(defrule AccessibilityPolicy1
(and (event (service ?service0&FlightService|Any) (type ?type0) (name ?event0&ProcessEvent))
(event (service ?service1&FlightService|Any) (type ?type1) (name ?event1&AccessibilityEvent)))
=>(service-state FlightService "Accessibility()") (assert (action (service ?service1) (name ?event1))))
```

PersonalHoliday Service Runtime Trace

```
#First request
Sending Event InputEvent
Sending Event ProcessEvent
[Login] Reading user list from file
Sending Event InputEvent
username = Susan.Smith
password = 1234
membership = FrequentFlyer
Sending Event ProcessEvent
[Login] Susan Smith has successfully login
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event InputEvent
Sending Event ProcessEvent
[Flight] Reading user list from file
[Flight] Reading airline list from file
Sending Event InputEvent
Sending Event ProcessEvent
Receiving Actions
Action: Accessibility
[Flight] Processing flight reservation request
[Flight] Searching for a flight from New York to Paris
```

```
[Flight] $$Reserving flight seat with Wheelchair access
[Flight] A flight was found for 300.0 euro
Sending Event OutputEvent
Sending Event IdleEvent
[Hotel] Reading user list from file
[Hotel] Reading hotel list from file
Sending Event InputEvent
Sending Event ProcessEvent
Receiving Actions
Action: Accessibility
[Hotel] Processing hotel reservation request
[Hotel] Searching for a hotel in Paris
[Hotel] Hotel found for the desired holiday location Hilton
[Hotel] $$Reserving room with Wheelchair access
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event InputEvent
Sending Event ProcessEvent
[HolidayBill] Processing Holiday Bill
[HolidayBill] Holiday package adds up to 2300.0
[HolidayBill] Sales tax is 10.0
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event OutputEvent
Sending Event IdleEvent
[Payment] Reading user list from file
Sending Event InputEvent
Sending Event ProcessEvent
[Payment] Processing Payment request for a bill of 2300.0
[Payment] Retrieving Visa Card Number
[Payment] Retrieving Visa Expiry Date
[Payment] Validating Visa Card
[Payment] Charging the user Susan Smith: 2530.0
[Payment] Payment of 2530.0 including sales tax of 10.0% has been approved
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event OutputEvent
Sending Event IdleEvent
```

```
#Second request
Sending Event InputEvent
Sending Event ProcessEvent
[Login] Reading user list from file
Sending Event InputEvent
username = John.Murphy
password = 1234
membership = SalesGroup
Sending Event ProcessEvent
[Login] John Murphy has successfully login
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event InputEvent
Sending Event ProcessEvent
[Flight] Reading user list from file
[Flight] Reading airline list from file
Sending Event InputEvent
Sending Event ProcessEvent
Receiving Actions
Action: BusinessClass
[Flight] Processing flight reservation request
```

```
[Flight] Searching for a flight from Toronto to London
[Flight] $$Reserving Business class flight
[Flight] A flight was found for 700.0 euro
Sending Event OutputEvent
Sending Event IdleEvent
[Hotel] Reading user list from file
[Hotel] Reading hotel list from file
Sending Event InputEvent
Sending Event ProcessEvent
Receiving Actions
Action: BusinessSuite
[Hotel] Processing hotel reservation request
[Hotel] Searching for a hotel in London
[Hotel] Hotel found for the desired holiday location Marriott
[Hotel] $$Reserving Business suite
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event InputEvent
Sending Event ProcessEvent
[HolidayBill] Processing Holiday Bill
[HolidayBill] Holiday package adds up to 1500.0
[HolidayBill] Sales tax is 10.0
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event OutputEvent
Sending Event IdleEvent
[Payment] Reading user list from file
Sending Event InputEvent
Sending Event ProcessEvent
[Payment] Processing Payment request for a bill of 1500.0
[Payment] Retrieving Visa Card Number
[Payment] Retrieving Visa Expiry Date
[Payment] Validating Visa Card
[Payment] Charging the user John Murphy: 1650.0
[Payment] Payment of 1650.0 including sales tax of 10.0% has been approved
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event OutputEvent
Sending Event IdleEvent
```

Appendix G – Notification Service Case Study

The artefacts produced for the Notification service case study. The artefacts are the description of the composite service, expressed as OWL-S; the description of the adaptive behaviours, expressed as a FSM; the high level management policies; and auto-generated refined low level enforceable policies. Lastly, the runtime trace of the web service managed by the refined policies.

Notification Service

```
<rdf:RDF
   xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
   xmlns:policy="http://kdeg.cs.tcd.ie/Policy.owl#"
   xmlns:thisprofile="http://www.daml.org/services/owl-s/1.1/NotificationProfile.owl#"
   xmlns="http://www.daml.org/services/owl-s/1.1/NotificationService.owl#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:base="http://www.daml.org/services/owl-s/1.1/NotificationService.owl"
   xmlns:thisgrounding="http://www.daml.org/services/owl-s/1.1/NotificationGrounding.owl#"
   xmlns:thisprocess="http://www.daml.org/services/owl-s/1.1/NotificationProcess.owl#"
   xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
   xmlns:grounding="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
  xml:base="http://www.daml.org/services/owl-s/1.1/NotificationService.owl">
  <owl:Ontology rdf:about="">
   <owl:versionInfo></owl:versionInfo>
   <owl:imports rdf:resource="Service.owl"/>
   <owl:imports rdf:resource="NotificationProfile.owl"/>
   <owl:imports rdf:resource="Profile.owl"/>
   <owl:imports rdf:resource="Grounding.owl"/>
   <owl:imports rdf:resource="Process.owl"/>
   <owl:imports rdf:resource="NotificationGrounding.owl"/>
   <owl:imports rdf:resource="NotificationProcess.owl"/>
   <rdfs:comment></rdfs:comment>
  </owl:Ontology>
  <service:Service rdf:ID="NotificationService">
   <service:describedBy rdf:resource="NotificationProcess.owl#NotificationService"/>
   <service:presents rdf:resource="NotificationProfile.owl#NotificationProfile"/>
   <service:supports rdf:resource="NotificationGrounding.owl#NotificationGrounding"/>
  </service:Service>
</rdf:RDF>
```

Notification Process

```
<rdf:RDF
   xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
   xmlns:objList="http://www.daml.org/services/owl-s/1.1/generic/ObjectList.owl#"
   xmlns:policy="http://kdeg.cs.tcd.ie/Policy.owl#"
```

```
  xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
  xmlns:base="http://www.daml.org/services/owl-s/1.1/NotificationProcess.owl"
  xmlns:grounding="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
  xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
  xmlns="http://www.daml.org/services/owl-s/1.1/NotificationProcess.owl#"
xml:base="http://www.daml.org/services/owl-s/1.1/NotificationProcess.owl">
<owl:Ontology rdf:about="">
  <owl:imports rdf:resource="NotificationProfile.owl"/>
  <owl:imports rdf:resource="Process.owl"/>
  <owl:imports rdf:resource="Service.owl"/>
  <owl:imports rdf:resource="Profile.owl"/>
  <owl:imports rdf:resource="NotificationService.owl"/>
  <owl:imports rdf:resource="NotificationGrounding.owl"/>
  <rdfs:comment></rdfs:comment>
  <owl:imports rdf:resource="Grounding.owl"/>
  <owl:versionInfo></owl:versionInfo>
</owl:Ontology>
<process:Output rdf:ID="UserContactNumber">
  <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
  >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
  <rdf:type rdf:resource="Process.owl#Input"/>
</process:Output>
<process:Perform rdf:ID="MessageServicePerform">
  <process:process>
    <process:AtomicProcess rdf:ID="MessageService">
      <process:hasInput>
        <process:Input rdf:ID="Subject">
          <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
        </process:Input>
      </process:hasInput>
      <process:hasInput>
        <process:Input rdf:ID="Message">
          <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
        </process:Input>
      </process:hasInput>
      <process:hasInput>
        <process:Input rdf:ID="Priority">
          <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
        </process:Input>
      </process:hasInput>
      <process:hasOutput>
        <process:Output rdf:ID="EmailMessage">
          <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
          <rdf:type rdf:resource="Process.owl#Input"/>
        </process:Output>
      </process:hasOutput>
      <process:hasOutput>
        <process:Output rdf:ID="VoiceMessage">
          <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
          <rdf:type rdf:resource="Process.owl#Input"/>
        </process:Output>
        </process:hasOutput>
        <process:hasOutput>
          <process:Output rdf:ID="Language">
            <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
            >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
          </process:Output>
        </process:hasOutput>
        <process:hasOutput>
          <process:Output rdf:ID="ProcessTime">
            <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
            >http://www.w3.org/2001/XMLSchema#time</process:parameterType>
          </process:Output>
        </process:hasOutput>
        <process:hasFiniteStateMachine
rdf:resource="MessageServiceFSM.owl#MessageServiceFSM"/>
      </process:AtomicProcess>
    </process:process>
    <process:hasDataFrom>
      <process:InputBinding>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#Subject"/>
            <process:fromProcess rdf:resource="#TheParentPerform"/>
          </process:ValueOf>
        </process:valueSource>
        <process:toParam rdf:resource="#Subject"/>
      </process:InputBinding>
    </process:hasDataFrom>
    <process:hasDataFrom>
      <process:InputBinding>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#Message"/>
            <process:fromProcess rdf:resource="#TheParentPerform"/>
          </process:ValueOf>
        </process:valueSource>
        <process:toParam rdf:resource="#Message"/>
      </process:InputBinding>
    </process:hasDataFrom>
    <process:hasDataFrom>
      <process:InputBinding>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#Priority"/>
            <process:fromProcess rdf:resource="#TheParentPerform"/>
          </process:ValueOf>
        </process:valueSource>
        <process:toParam rdf:resource="#Priority"/>
      </process:InputBinding>
    </process:hasDataFrom>
    <process:hasDataFrom>
      <process:InputBinding>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#ProcessTime"/>
            <process:fromProcess rdf:resource="#TheParentPerform"/>
          </process:ValueOf>
        </process:valueSource>
        <process:toParam rdf:resource="#ProcessTime"/>
      </process:InputBinding>
    </process:hasDataFrom>
  </process:Perform>
  <process:AtomicProcess rdf:ID="AddressBookService">
    <process:hasOutput>
      <process:Output rdf:ID="RecipientContactNumber">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
        <rdf:type rdf:resource="Process.owl#Input"/>
      </process:Output>
    </process:hasOutput>
    <process:hasOutput rdf:resource="#UserContactNumber"/>
    <process:hasInput>
      <process:Input rdf:ID="Department">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
      </process:Input>
    </process:hasInput>
    <process:hasInput>
      <process:Input rdf:ID="Recipient">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
      </process:Input>
    </process:hasInput>
    <process:hasOutput>
      <process:Output rdf:ID="UserEmailAddress">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
        <rdf:type rdf:resource="Process.owl#Input"/>
      </process:Output>
    </process:hasOutput>
    <process:hasOutput>
      <process:Output rdf:ID="SearchTime">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/2001/XMLSchema#time</process:parameterType>
      </process:Output>
    </process:hasOutput>
    <process:hasFiniteStateMachine
rdf:resource="AddressBookServiceFSM.owl#AddressBookServiceFSM"/>
    <process:hasOutput>
      <process:Output rdf:ID="RecipientEmailAddress">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
        <rdf:type rdf:resource="Process.owl#Input"/>
      </process:Output>
    </process:hasOutput>
    <process:hasInput>
      <process:Output rdf:ID="Guestid">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
        <rdf:type rdf:resource="Process.owl#Input"/>
      </process:Output>
    </process:hasInput>
  </process:AtomicProcess>
  <process:Input rdf:ID="Membership">
    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
  </process:Input>
  <process:Input rdf:ID="RecipientContacNumber">
    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
```

```
</process:Input>
<process:Perform rdf:ID="PhoneServicePerform">
  <process:process>
    <process:AtomicProcess rdf:ID="PhoneService">
      <process:hasInput rdf:resource="#UserContactNumber"/>
      <process:hasInput rdf:resource="#RecipientContactNumber"/>
      <process:hasInput rdf:resource="#VoiceMessage"/>
      <process:hasOutput>
        <process:Output rdf:ID="CallResponse">
          <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
        </process:Output>
      </process:hasOutput>
      <process:hasFiniteStateMachine rdf:resource="PhoneServiceFSM.owl#PhoneServiceFSM"/>
    </process:AtomicProcess>
  </process:process>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#UserContactNumber"/>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#UserContactNumber"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#RecipientContactNumber"/>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#RecipientContactNumber"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#VoiceMessage"/>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#VoiceMessage"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#CallResponse"/>
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#CallResponse"/>
    </process:InputBinding>
  </process:hasDataFrom>
```

```
</process:Perform>
<process:Perform rdf:ID="ContactServicePerform">
 <process:hasDataFrom>
  <process:InputBinding>
   <process:valueSource>
    <process:ValueOf>
     <process:theVar rdf:resource="#RecipientEmailAddress"/>
     <process:fromProcess>
      <process:Perform rdf:ID="AddressBookServicePerform">
       <process:process rdf:resource="#AddressBookService"/>
       <process:hasDataFrom>
        <process:InputBinding>
         <process:valueSource>
          <process:ValueOf>
           <process:theVar rdf:resource="#Recipient"/>
           <process:fromProcess rdf:resource="#TheParentPerform"/>
          </process:ValueOf>
         </process:valueSource>
         <process:toParam rdf:resource="#Recipient"/>
        </process:InputBinding>
       </process:hasDataFrom>
       <process:hasDataFrom>
        <process:InputBinding>
         <process:valueSource>
          <process:ValueOf>
           <process:theVar rdf:resource="#Department"/>
           <process:fromProcess rdf:resource="#TheParentPerform"/>
          </process:ValueOf>
         </process:valueSource>
         <process:toParam rdf:resource="#Department"/>
        </process:InputBinding>
       </process:hasDataFrom>
       <process:hasDataFrom>
        <process:InputBinding>
         <process:valueSource>
          <process:ValueOf>
           <process:theVar rdf:resource="#Guestid"/>
           <process:fromProcess>
            <process:Perform rdf:ID="LoginServicePerform">
             <process:process>
              <process:AtomicProcess rdf:ID="LoginService">
               <process:hasInput>
                <process:Input rdf:ID="Username">
                 <process:parameterType rdf:datatype=
                 "http://www.w3.org/2001/XMLSchema#anyURI"
                 >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
                </process:Input>
               </process:hasInput>
               <process:hasInput>
                <process:Input rdf:ID="Password">
                 <process:parameterType rdf:datatype=
                 "http://www.w3.org/2001/XMLSchema#anyURI"
                 >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
                </process:Input>
               </process:hasInput>
               <process:hasInput rdf:resource="#Membership"/>
               <process:hasOutput rdf:resource="#Guestid"/>
              </process:AtomicProcess>
             </process:process>
             <process:hasDataFrom>
```

```
                          <process:InputBinding>
                            <process:valueSource>
                              <process:ValueOf>
                                <process:theVar rdf:resource="#Username"/>
                                <process:fromProcess rdf:resource="#TheParentPerform"/>
                              </process:ValueOf>
                            </process:valueSource>
                            <process:toParam rdf:resource="#Username"/>
                          </process:InputBinding>
                        </process:hasDataFrom>
                        <process:hasDataFrom>
                          <process:InputBinding>
                            <process:valueSource>
                              <process:ValueOf>
                                <process:theVar rdf:resource="#Password"/>
                                <process:fromProcess rdf:resource="#TheParentPerform"/>
                              </process:ValueOf>
                            </process:valueSource>
                            <process:toParam rdf:resource="#Password"/>
                          </process:InputBinding>
                        </process:hasDataFrom>
                        <process:hasDataFrom>
                          <process:InputBinding>
                            <process:valueSource>
                              <process:ValueOf>
                                <process:theVar rdf:resource="#Membership"/>
                                <process:fromProcess rdf:resource="#TheParentPerform"/>
                              </process:ValueOf>
                            </process:valueSource>
                            <process:toParam rdf:resource="#Membership"/>
                          </process:InputBinding>
                        </process:hasDataFrom>
                      </process:Perform>
                    </process:fromProcess>
                  </process:ValueOf>
                </process:valueSource>
                <process:toParam rdf:resource="#Guestid"/>
              </process:InputBinding>
            </process:hasDataFrom>
            <process:hasDataFrom>
              <process:InputBinding>
                <process:valueSource>
                  <process:ValueOf>
                    <process:theVar rdf:resource="#SearchTime"/>
                    <process:fromProcess rdf:resource="#TheParentPerform"/>
                  </process:ValueOf>
                </process:valueSource>
                <process:toParam rdf:resource="#SearchTime"/>
              </process:InputBinding>
            </process:hasDataFrom>
          </process:Perform>
        </process:fromProcess>
      </process:ValueOf>
    </process:valueSource>
    <process:toParam rdf:resource="#RecipientEmailAddress"/>
  </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
  <process:InputBinding>
    <process:valueSource>
```

```
      <process:ValueOf>
        <process:theVar rdf:resource="#EmailMessage"/>
        <process:fromProcess rdf:resource="#ThisPerform"/>
      </process:ValueOf>
    </process:valueSource>
    <process:toParam rdf:resource="#EmailMessage"/>
  </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
  <process:InputBinding>
    <process:valueSource>
      <process:ValueOf>
        <process:theVar rdf:resource="#VoiceMessage"/>
        <process:fromProcess rdf:resource="#MessageServicePerform"/>
      </process:ValueOf>
    </process:valueSource>
    <process:toParam rdf:resource="#VoiceMessage"/>
  </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
  <process:InputBinding>
    <process:valueSource>
      <process:ValueOf>
        <process:theVar rdf:resource="#VoiceMessage"/>
        <process:fromProcess rdf:resource="#ThisPerform"/>
      </process:ValueOf>
    </process:valueSource>
    <process:toParam rdf:resource="#VoiceMessage"/>
  </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
  <process:InputBinding>
    <process:valueSource>
      <process:ValueOf>
        <process:theVar rdf:resource="#CallResponse"/>
        <process:fromProcess rdf:resource="#TheParentPerform"/>
      </process:ValueOf>
    </process:valueSource>
    <process:toParam rdf:resource="#CallResponse"/>
  </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
  <process:InputBinding>
    <process:valueSource>
      <process:ValueOf>
        <process:theVar rdf:resource="#UserEmailAddress"/>
        <process:fromProcess rdf:resource="#ThisPerform"/>
      </process:ValueOf>
    </process:valueSource>
    <process:toParam rdf:resource="#UserEmailAddress"/>
  </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
  <process:InputBinding>
    <process:valueSource>
      <process:ValueOf>
        <process:theVar rdf:resource="#UserContactNumber"/>
        <process:fromProcess rdf:resource="#AddressBookServicePerform"/>
      </process:ValueOf>
    </process:valueSource>
```

```
      <process:toParam rdf:resource="#UserContactNumber"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:process>
    <process:CompositeProcess rdf:ID="ContactService">
      <process:hasFiniteStateMachine rdf:resource="ContactServiceFSM.owl#ContactServiceFSM"/>
      <process:hasInput>
        <process:Input rdf:ID="UserEmailAdress">
          <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
        </process:Input>
      </process:hasInput>
      <process:hasOutput rdf:resource="#CallResponse"/>
      <process:hasInput rdf:resource="#RecipientEmailAddress"/>
      <process:hasInput rdf:resource="#EmailMessage"/>
      <process:hasInput rdf:resource="#UserContactNumber"/>
      <process:hasOutput>
        <process:Output rdf:ID="EmailResponse">
          <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
        </process:Output>
      </process:hasOutput>
      <process:hasInput rdf:resource="#RecipientContacNumber"/>
      <process:composedOf>
        <process:Sequence>
          <process:components>
            <process:ControlConstructList>
              <objList:first rdf:resource="#PhoneServicePerform"/>
              <objList:rest>
                <process:ControlConstructList>
                  <objList:first>
                    <process:Perform rdf:ID="EmailServicePerform">
                      <process:process>
                        <process:AtomicProcess rdf:ID="EmailService">
                          <process:hasInput rdf:resource="#UserEmailAddress"/>
                          <process:hasInput rdf:resource="#RecipientEmailAddress"/>
                          <process:hasInput rdf:resource="#EmailMessage"/>
                          <process:hasOutput rdf:resource="#EmailResponse"/>
                          <process:hasFiniteStateMachine
rdf:resource="EmailServiceFSM.owl#EmailServiceFSM"/>
                        </process:AtomicProcess>
                      </process:process>
                      <process:hasDataFrom>
                        <process:InputBinding>
                          <process:valueSource>
                            <process:ValueOf>
                              <process:theVar rdf:resource="#UserEmailAddress"/>
                              <process:fromProcess rdf:resource="#TheParentPerform"/>
                            </process:ValueOf>
                          </process:valueSource>
                          <process:toParam rdf:resource="#UserEmailAddress"/>
                        </process:InputBinding>
                      </process:hasDataFrom>
                      <process:hasDataFrom>
                        <process:InputBinding>
                          <process:valueSource>
                            <process:ValueOf>
                              <process:theVar rdf:resource="#RecipientEmailAddress"/>
                              <process:fromProcess rdf:resource="#TheParentPerform"/>
                            </process:ValueOf>
            </process:valueSource>
            <process:toParam rdf:resource="#RecipientEmailAddress"/>
           </process:InputBinding>
          </process:hasDataFrom>
          <process:hasDataFrom>
           <process:InputBinding>
            <process:valueSource>
             <process:ValueOf>
              <process:theVar rdf:resource="#EmailMessage"/>
              <process:fromProcess rdf:resource="#TheParentPerform"/>
             </process:ValueOf>
            </process:valueSource>
            <process:toParam rdf:resource="#EmailMessage"/>
           </process:InputBinding>
          </process:hasDataFrom>
          <process:hasDataFrom>
           <process:InputBinding>
            <process:valueSource>
             <process:ValueOf>
              <process:theVar rdf:resource="#EmailResponse"/>
              <process:fromProcess rdf:resource="#TheParentPerform"/>
             </process:ValueOf>
            </process:valueSource>
            <process:toParam rdf:resource="#EmailResponse"/>
           </process:InputBinding>
          </process:hasDataFrom>
         </process:Perform>
        </objList:first>
        <objList:rest rdf:resource="generic/ObjectList.owl#nil"/>
       </process:ControlConstructList>
      </objList:rest>
     </process:ControlConstructList>
    </process:components>
   </process:Sequence>
  </process:composedOf>
  <process:hasInput rdf:resource="#VoiceMessage"/>
 </process:CompositeProcess>
</process:process>
<process:hasDataFrom>
 <process:InputBinding>
  <process:valueSource>
   <process:ValueOf>
    <process:theVar rdf:resource="#CallResponse"/>
    <process:fromProcess rdf:resource="#ThisPerform"/>
   </process:ValueOf>
  </process:valueSource>
  <process:toParam rdf:resource="#CallResponse"/>
 </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
 <process:InputBinding>
  <process:valueSource>
   <process:ValueOf>
    <process:theVar rdf:resource="#RecipientContactNumber"/>
    <process:fromProcess rdf:resource="#ThisPerform"/>
   </process:ValueOf>
  </process:valueSource>
  <process:toParam rdf:resource="#RecipientContactNumber"/>
 </process:InputBinding>
</process:hasDataFrom>
```

```
<process:hasDataFrom>
 <process:InputBinding>
  <process:valueSource>
   <process:ValueOf>
    <process:theVar rdf:resource="#EmailResponse"/>
    <process:fromProcess rdf:resource="#ThisPerform"/>
   </process:ValueOf>
  </process:valueSource>
  <process:toParam rdf:resource="#EmailResponse"/>
 </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
 <process:InputBinding>
  <process:valueSource>
   <process:ValueOf>
    <process:theVar rdf:resource="#RecipientContactNumber"/>
    <process:fromProcess rdf:resource="#AddressBookServicePerform"/>
   </process:ValueOf>
  </process:valueSource>
  <process:toParam rdf:resource="#RecipientContacNumber"/>
 </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
 <process:InputBinding>
  <process:valueSource>
   <process:ValueOf>
    <process:theVar rdf:resource="#EmailMessage"/>
    <process:fromProcess rdf:resource="#MessageServicePerform"/>
   </process:ValueOf>
  </process:valueSource>
  <process:toParam rdf:resource="#EmailMessage"/>
 </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
 <process:InputBinding>
  <process:valueSource>
   <process:ValueOf>
    <process:theVar rdf:resource="#UserEmailAddress"/>
    <process:fromProcess rdf:resource="#AddressBookServicePerform"/>
   </process:ValueOf>
  </process:valueSource>
  <process:toParam rdf:resource="#UserEmailAdress"/>
 </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
 <process:InputBinding>
  <process:valueSource>
   <process:ValueOf>
    <process:theVar rdf:resource="#RecipientEmailAddress"/>
    <process:fromProcess rdf:resource="#ThisPerform"/>
   </process:ValueOf>
  </process:valueSource>
  <process:toParam rdf:resource="#RecipientEmailAddress"/>
 </process:InputBinding>
</process:hasDataFrom>
<process:hasDataFrom>
 <process:InputBinding>
  <process:valueSource>
   <process:ValueOf>
    <process:theVar rdf:resource="#EmailResponse"/>
```

```
          <process:fromProcess rdf:resource="#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#EmailResponse"/>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#UserContactNumber"/>
          <process:fromProcess rdf:resource="#ThisPerform"/>
        </process:ValueOf>
      </process:valueSource>
      <process:toParam rdf:resource="#UserContactNumber"/>
    </process:InputBinding>
  </process:hasDataFrom>
</process:Perform>
<process:CompositeProcess rdf:ID="NotificationService">
  <process:hasInput rdf:resource="#Priority"/>
  <process:hasOutput rdf:resource="#ProcessTime"/>
  <process:hasOutput rdf:resource="#Language"/>
  <process:hasInput rdf:resource="#Subject"/>
  <process:hasInput rdf:resource="#Department"/>
  <process:hasInput rdf:resource="#Message"/>
  <process:hasInput rdf:resource="#Recipient"/>
  <process:hasOutput rdf:resource="#SearchTime"/>
  <process:hasFiniteStateMachine
rdf:resource="NotificationServiceFSM.owl#NotificationServiceFSM"/>
  <process:hasInput rdf:resource="#Username"/>
  <process:hasOutput rdf:resource="#CallResponse"/>
  <process:hasInput rdf:resource="#Password"/>
  <process:composedOf>
    <process:Sequence>
      <process:components>
        <process:ControlConstructList>
          <objList:first rdf:resource="#LoginServicePerform"/>
          <objList:rest>
            <process:ControlConstructList>
              <objList:first rdf:resource="#MessageServicePerform"/>
              <objList:rest>
                <process:ControlConstructList>
                  <objList:first rdf:resource="#AddressBookServicePerform"/>
                  <objList:rest>
                    <process:ControlConstructList>
                      <objList:first rdf:resource="#ContactServicePerform"/>
                      <objList:rest rdf:resource="generic/ObjectList.owl#nil"/>
                    </process:ControlConstructList>
                  </objList:rest>
                </process:ControlConstructList>
              </objList:rest>
            </process:ControlConstructList>
          </objList:rest>
        </process:ControlConstructList>
      </process:components>
    </process:Sequence>
  </process:composedOf>
  <process:hasOutput rdf:resource="#EmailResponse"/>
  <process:hasInput rdf:resource="#Membership"/>
</process:CompositeProcess>
```

```
</rdf:RDF>
```

Notification Grounding

```
<rdf:RDF
  xmlns:wsdl="http://www.daml.org/services/owl-s/1.1/NotificationGrounding.wsdl#"
  xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
  xmlns:policy="http://kdeg.cs.tcd.ie/Policy.owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:base="http://www.daml.org/services/owl-s/1.1/NotificationGrounding.owl"
  xmlns="http://www.daml.org/services/owl-s/1.1/NotificationGrounding.owl#"
  xmlns:thisprocess="http://www.daml.org/services/owl-s/1.1/NotificationProcess.owl#"
  xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
  xmlns:grounding="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:wsdldoc="http://www.daml.org/services/owl-s/1.1/NotificationGrounding.wsdl"
  xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
 xml:base="http://www.daml.org/services/owl-s/1.1/NotificationGrounding.owl">
 <owl:Ontology rdf:about="">
  <owl:versionInfo></owl:versionInfo>
  <rdfs:comment></rdfs:comment>
  <owl:imports rdf:resource="Service.owl"/>
  <owl:imports rdf:resource="Process.owl"/>
  <owl:imports rdf:resource="Grounding.owl"/>
  <owl:imports rdf:resource="NotificationProcess.owl"/>
 </owl:Ontology>
 <grounding:WsdlAtomicProcessGrounding rdf:ID="MessageServiceGrounding">
  <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
  >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:otherReference>
  <grounding:wsdlDocument rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
  >http://www.daml.org/services/owl-s/1.1/NotificationGrounding.wsdl</grounding:wsdlDocument>
  <grounding:wsdlInputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
  >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#MessageService_Input</grounding:wsdlInputMessage>
  <grounding:wsdlOutputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
  >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#MessageService_Output</grounding:wsdlOutputMessage>
  <grounding:wsdlInput>
   <grounding:WsdlInputMessageMap>
    <grounding:owlsParameter rdf:resource="NotificationProcess.owl#Message"/>
    <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#message</grounding:wsdlMessagePart>
   </grounding:WsdlInputMessageMap>
  </grounding:wsdlInput>
  <grounding:wsdlReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
  >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:wsdlReference>
  <grounding:wsdlOperation>
   <grounding:WsdlOperationRef>
    <grounding:portType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#MessageService_PortType</grounding:portType>
    <grounding:operation rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#MessageService_Operation</grounding:operation>
   </grounding:WsdlOperationRef>
  </grounding:wsdlOperation>
  <grounding:wsdlOutput>
```

```
      <grounding:WsdlOutputMessageMap>
        <grounding:owlsParameter rdf:resource="NotificationProcess.owl#Language"/>
        <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#lang</grounding:wsdlMessagePart>
      </grounding:WsdlOutputMessageMap>
    </grounding:wsdlOutput>
    <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://schemas.wmlsoap.org/soap/http/</grounding:otherReference>
    <grounding:wsdlOutput>
      <grounding:WsdlOutputMessageMap>
        <grounding:owlsParameter rdf:resource="NotificationProcess.owl#EmailMessage"/>
        <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#emailmsg</grounding:wsdlMessagePart>
      </grounding:WsdlOutputMessageMap>
    </grounding:wsdlOutput>
    <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://schemas.wmlsoap.org/wsdl/soap/</grounding:otherReference>
    <grounding:wsdlOutput>
      <grounding:WsdlOutputMessageMap>
        <grounding:owlsParameter rdf:resource="NotificationProcess.owl#ProcessTime"/>
        <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#time</grounding:wsdlMessagePart>
      </grounding:WsdlOutputMessageMap>
    </grounding:wsdlOutput>
    <grounding:wsdlInput>
      <grounding:WsdlInputMessageMap>
        <grounding:owlsParameter rdf:resource="NotificationProcess.owl#Subject"/>
        <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#subject</grounding:wsdlMessagePart>
      </grounding:WsdlInputMessageMap>
    </grounding:wsdlInput>
    <grounding:owlsProcess rdf:resource="NotificationProcess.owl#MessageService"/>
    <grounding:wsdlInput>
      <grounding:WsdlInputMessageMap>
        <grounding:owlsParameter rdf:resource="NotificationProcess.owl#Priority"/>
        <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#priority</grounding:wsdlMessagePart>
      </grounding:WsdlInputMessageMap>
    </grounding:wsdlInput>
    <grounding:wsdlOutput>
      <grounding:WsdlOutputMessageMap>
        <grounding:owlsParameter rdf:resource="NotificationProcess.owl#VoiceMessage"/>
        <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#voicemsg</grounding:wsdlMessagePart>
      </grounding:WsdlOutputMessageMap>
    </grounding:wsdlOutput>
  </grounding:WsdlAtomicProcessGrounding>
  <grounding:WsdlGrounding rdf:ID="NotificationGrounding">
    <grounding:hasAtomicProcessGrounding>
      <grounding:WsdlAtomicProcessGrounding rdf:ID="LoginServiceGrounding">
        <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:otherReference>
        <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://schemas.wmlsoap.org/soap/http/</grounding:otherReference>
        <grounding:wsdlReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:wsdlReference>
        <grounding:wsdlInput>
          <grounding:WsdlInputMessageMap>
            <grounding:owlsParameter rdf:resource="NotificationProcess.owl#Username"/>
            <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
            >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#name</grounding:wsdlMessagePart>
          </grounding:WsdlInputMessageMap>
        </grounding:wsdlInput>
        <grounding:wsdlInput>
          <grounding:WsdlInputMessageMap>
            <grounding:owlsParameter rdf:resource="NotificationProcess.owl#Membership"/>
            <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
            >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#member</grounding:wsdlMessagePart>
          </grounding:WsdlInputMessageMap>
        </grounding:wsdlInput>
        <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://schemas.wmlsoap.org/wsdl/soap/</grounding:otherReference>
        <grounding:wsdlOutputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#LoginService_Output</grounding:wsdlOutputMessage>
        <grounding:wsdlInput>
          <grounding:WsdlInputMessageMap>
            <grounding:owlsParameter rdf:resource="NotificationProcess.owl#Password"/>
            <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
            >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#pass</grounding:wsdlMessagePart>
          </grounding:WsdlInputMessageMap>
        </grounding:wsdlInput>
        <grounding:wsdlOutput>
          <grounding:WsdlOutputMessageMap>
            <grounding:owlsParameter rdf:resource="NotificationProcess.owl#Guestid"/>
            <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
            >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#user</grounding:wsdlMessagePart>
          </grounding:WsdlOutputMessageMap>
        </grounding:wsdlOutput>
        <grounding:wsdlInputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#LoginService_Input</grounding:wsdlInputMessage>
        <grounding:wsdlDocument rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl</grounding:wsdlDocument>
        <grounding:wsdlOperation>
          <grounding:WsdlOperationRef>
            <grounding:portType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
            >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#LoginService_PortType</grounding:portType>
            <grounding:operation rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
            >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#LoginService_Operation</grounding:operation>
          </grounding:WsdlOperationRef>
        </grounding:wsdlOperation>
        <grounding:owlsProcess rdf:resource="NotificationProcess.owl#LoginService"/>
      </grounding:WsdlAtomicProcessGrounding>
    </grounding:hasAtomicProcessGrounding>
    <grounding:hasAtomicProcessGrounding rdf:resource="#MessageServiceGrounding"/>
    <grounding:hasAtomicProcessGrounding>
    <grounding:WsdlAtomicProcessGrounding rdf:ID="AddressBookServiceGrounding">
      <grounding:wsdlOutputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#AddressBookService_Output</grounding:wsdlOutputMessage>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="NotificationProcess.owl#Department"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#group</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
          <grounding:owlsParameter rdf:resource="NotificationProcess.owl#UserEmailAddress"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#fromemail</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
      </grounding:wsdlOutput>
      <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:otherReference>
      <grounding:wsdlOperation>
        <grounding:WsdlOperationRef>
          <grounding:portType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#AddressBookService_PortType</grounding:portType>
          <grounding:operation rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#AddressBookService_Operation</grounding:operation>
        </grounding:WsdlOperationRef>
      </grounding:wsdlOperation>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="NotificationProcess.owl#Guestid"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#user</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
          <grounding:owlsParameter rdf:resource="NotificationProcess.owl#RecipientEmailAddress"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#toemail</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
      </grounding:wsdlOutput>
      <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
          <grounding:owlsParameter rdf:resource="NotificationProcess.owl#SearchTime"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#time</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
      </grounding:wsdlOutput>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="NotificationProcess.owl#Recipient"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#recipient</grounding:wsdlMessagePart>
      </grounding:WsdlInputMessageMap>
    </grounding:wsdlInput>
    <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://schemas.wmlsoap.org/soap/http/</grounding:otherReference>
    <grounding:wsdlOutput>
      <grounding:WsdlOutputMessageMap>
        <grounding:owlsParameter
rdf:resource="NotificationProcess.owl#RecipientContactNumber"/>
        <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#callee</grounding:wsdlMessagePart>
      </grounding:WsdlOutputMessageMap>
    </grounding:wsdlOutput>
    <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://schemas.wmlsoap.org/wsdl/soap/</grounding:otherReference>
    <grounding:wsdlDocument rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl</grounding:wsdlDocument>
    <grounding:wsdlOutput>
      <grounding:WsdlOutputMessageMap>
        <grounding:owlsParameter rdf:resource="NotificationProcess.owl#UserContactNumber"/>
        <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#caller</grounding:wsdlMessagePart>
      </grounding:WsdlOutputMessageMap>
    </grounding:wsdlOutput>
    <grounding:wsdlInputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#AddressBookService_Input</grounding:wsdlInputMessage>
    <grounding:wsdlReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:wsdlReference>
    <grounding:owlsProcess rdf:resource="NotificationProcess.owl#AddressBookService"/>
  </grounding:WsdlAtomicProcessGrounding>
</grounding:hasAtomicProcessGrounding>
<grounding:hasAtomicProcessGrounding>
  <grounding:WsdlAtomicProcessGrounding rdf:ID="PhoneServiceGrounding">
    <grounding:wsdlOutput>
      <grounding:WsdlOutputMessageMap>
        <grounding:owlsParameter rdf:resource="NotificationProcess.owl#CallResponse"/>
        <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#response</grounding:wsdlMessagePart>
      </grounding:WsdlOutputMessageMap>
    </grounding:wsdlOutput>
    <grounding:owlsProcess rdf:resource="NotificationProcess.owl#PhoneService"/>
    <grounding:wsdlOperation>
      <grounding:WsdlOperationRef>
        <grounding:portType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#PhoneService_PortType</grounding:portType>
        <grounding:operation rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#PhoneService_Operation</grounding:operation>
      </grounding:WsdlOperationRef>
    </grounding:wsdlOperation>
    <grounding:wsdlInput>
      <grounding:WsdlInputMessageMap>
        <grounding:owlsParameter rdf:resource="NotificationProcess.owl#VoiceMessage"/>
```

```
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#message</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:wsdlOutputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#PhoneService_Output</grounding:wsdlOutputMessage>
      <grounding:wsdlInputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#PhoneService_Input</grounding:wsdlInputMessage>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter
rdf:resource="NotificationProcess.owl#RecipientContactNumber"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#callee</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:wsdlDocument rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl</grounding:wsdlDocument>
      <grounding:wsdlReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:wsdlReference>
      <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:otherReference>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="NotificationProcess.owl#UserContactNumber"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#caller</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://schemas.wmlsoap.org/wsdl/soap/</grounding:otherReference>
      <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://schemas.wmlsoap.org/soap/http/</grounding:otherReference>
    </grounding:WsdlAtomicProcessGrounding>
  </grounding:hasAtomicProcessGrounding>
  <grounding:hasAtomicProcessGrounding>
    <grounding:WsdlAtomicProcessGrounding rdf:ID="EmailServiceGrounding">
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="NotificationProcess.owl#EmailMessage"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#message</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:wsdlDocument rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl</grounding:wsdlDocument>
      <grounding:owlsProcess rdf:resource="NotificationProcess.owl#EmailService"/>
      <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://schemas.wmlsoap.org/soap/http/</grounding:otherReference>
      <grounding:wsdlReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:wsdlReference>
      <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://schemas.wmlsoap.org/wsdl/soap/</grounding:otherReference>
      <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
          <grounding:owlsParameter rdf:resource="NotificationProcess.owl#EmailResponse"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#response</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
      </grounding:wsdlOutput>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="NotificationProcess.owl#UserEmailAddress"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#fromemail</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:wsdlOutputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#EmailService_Output</grounding:wsdlOutputMessage>
      <grounding:otherReference rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.w3.org/TR/2001/NOTE-wsdl-20010315</grounding:otherReference>
      <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
          <grounding:owlsParameter rdf:resource="NotificationProcess.owl#RecipientEmailAddress"/>
          <grounding:wsdlMessagePart rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#toemail</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
      </grounding:wsdlInput>
      <grounding:wsdlInputMessage rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
      >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#EmailService_Input</grounding:wsdlInputMessage>
      <grounding:wsdlOperation>
        <grounding:WsdlOperationRef>
          <grounding:portType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#EmailService_PortType</grounding:portType>
          <grounding:operation rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.daml.org/services/owl-
s/1.1/NotificationGrounding.wsdl#EmailService_Operation</grounding:operation>
        </grounding:WsdlOperationRef>
      </grounding:wsdlOperation>
    </grounding:WsdlAtomicProcessGrounding>
  </grounding:hasAtomicProcessGrounding>
 </grounding:WsdlGrounding>
</rdf:RDF>
```

Notification Service's FSM

```
<rdf:RDF
  xmlns:fsm="http://kdeg.cs.tcd.ie/FiniteStateMachine.owl#"
  xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
  xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
  xmlns:policy="http://kdeg.cs.tcd.ie/Policy.owl#"
  xmlns:grounding="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
  xmlns="http://www.daml.org/services/owl-s/1.1/NotificationServiceFSM.owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:base="http://www.daml.org/services/owl-s/1.1/NotificationServiceFSM.owl"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
  xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
xml:base="http://www.daml.org/services/owl-s/1.1/NotificationServiceFSM.owl">
<owl:Ontology rdf:about="">
  <owl:imports rdf:resource="http://kdeg.cs.tcd.ie/FiniteStateMachine.owl"/>
</owl:Ontology>
<fsm:SubmachineState rdf:ID="RichContentSubSM">
  <fsm:submachine>
    <fsm:StateMachine rdf:ID="RichContent">
      <fsm:top>
        <fsm:CompositeState rdf:ID="RichContentCS">
          <fsm:subvertex>
            <fsm:PseudoState rdf:ID="RichContentInitialState">
              <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
              <fsm:outgoing>
                <fsm:Transition rdf:ID="RichContentInitialStateInitialTransition">
                  <fsm:trigger>
                    <fsm:SignalEvent rdf:ID="InitialEvent">
                      <fsm:signal>
                        <fsm:Signal rdf:ID="InitialEventSource">
                          <fsm:value>InitialSignal</fsm:value>
                        </fsm:Signal>
                      </fsm:signal>
                    </fsm:SignalEvent>
                  </fsm:trigger>
                  <fsm:source rdf:resource="#RichContentInitialState"/>
                  <fsm:target>
                    <fsm:SimpleState rdf:ID="RichContentState1">
                      <fsm:doActivity>doHtmlMessage()</fsm:doActivity>
                      <fsm:incoming rdf:resource="#RichContentInitialStateInitialTransition"/>
                      <fsm:outgoing>
                        <fsm:Transition rdf:ID="RichContentState1Transition1">
                          <fsm:trigger>
                            <fsm:SignalEvent rdf:ID="MessageCreatedEvent">
                              <fsm:signal>
                                <fsm:Signal rdf:ID="MessageCreatedEventSource">
                                  <fsm:value>MessageCreated</fsm:value>
                                </fsm:Signal>
                              </fsm:signal>
                            </fsm:SignalEvent>
                          </fsm:trigger>
                          <fsm:source rdf:resource="#RichContentState1"/>
                          <fsm:target>
                            <fsm:FinalState rdf:ID="RichContentState2">
                              <fsm:doActivity>doHmtlHeader()</fsm:doActivity>
                              <fsm:incoming rdf:resource="#RichContentState1Transition1"/>
                            </fsm:FinalState>
                          </fsm:target>
                        </fsm:Transition>
                      </fsm:outgoing>
                    </fsm:SimpleState>
                  </fsm:target>
                </fsm:Transition>
              </fsm:outgoing>
            </fsm:PseudoState>
          </fsm:subvertex>
          <fsm:subvertex rdf:resource="#RichContentState1"/>
          <fsm:subvertex rdf:resource="#RichContentState2"/>
        </fsm:CompositeState>
      </fsm:top>
      <fsm:comment>Enriches the email content with HTML format</fsm:comment>
    </fsm:StateMachine>
  </fsm:submachine>
</fsm:SubmachineState>
<fsm:FinalState rdf:ID="AuthenticationState1">
  <fsm:doActivity>setAuthentication()</fsm:doActivity>
  <fsm:incoming>
    <fsm:Transition rdf:ID="AuthenticationInitialStateInitialTransition">
      <fsm:trigger rdf:resource="#InitialEvent"/>
      <fsm:source>
        <fsm:PseudoState rdf:ID="AuthenticationInitialState">
          <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
          <fsm:outgoing rdf:resource="#AuthenticationInitialStateInitialTransition"/>
        </fsm:PseudoState>
      </fsm:source>
      <fsm:target rdf:resource="#AuthenticationState1"/>
    </fsm:Transition>
  </fsm:incoming>
</fsm:FinalState>
<fsm:Transition rdf:ID="QuickSearchInitialStateInitialTransition">
  <fsm:trigger rdf:resource="#InitialEvent"/>
  <fsm:source>
    <fsm:PseudoState rdf:ID="QuickSearchInitialState">
      <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
      <fsm:outgoing rdf:resource="#QuickSearchInitialStateInitialTransition"/>
    </fsm:PseudoState>
  </fsm:source>
  <fsm:target>
    <fsm:SimpleState rdf:ID="QuickSearchState1">
      <fsm:doActivity>doQuickSort()</fsm:doActivity>
      <fsm:incoming rdf:resource="#QuickSearchInitialStateInitialTransition"/>
      <fsm:outgoing>
        <fsm:Transition rdf:ID="QuickSearchState1Transition1">
          <fsm:trigger>
            <fsm:SignalEvent rdf:ID="SortedEvent">
              <fsm:signal>
                <fsm:Signal rdf:ID="SortedEventSource">
                  <fsm:value>Sorted</fsm:value>
                </fsm:Signal>
              </fsm:signal>
            </fsm:SignalEvent>
          </fsm:trigger>
          <fsm:source rdf:resource="#QuickSearchState1"/>
          <fsm:target>
            <fsm:SimpleState rdf:ID="QuickSearchState2">
              <fsm:doActivity>doBinarySearch()</fsm:doActivity>
              <fsm:incoming rdf:resource="#QuickSearchState1Transition1"/>
            </fsm:SimpleState>
          </fsm:target>
        </fsm:Transition>
      </fsm:outgoing>
    </fsm:SimpleState>
  </fsm:target>
</fsm:Transition>
<fsm:CompositeState rdf:ID="ProcessState">
  <fsm:subvertex>
    <fsm:SubmachineState rdf:ID="QuickSearchSubSM">
      <fsm:submachine>
      <fsm:StateMachine rdf:ID="QuickSearch">
        <fsm:top>
          <fsm:CompositeState rdf:ID="QuickSearchCS">
            <fsm:subvertex rdf:resource="#QuickSearchInitialState"/>
            <fsm:subvertex rdf:resource="#QuickSearchState1"/>
            <fsm:subvertex rdf:resource="#QuickSearchState2"/>
          </fsm:CompositeState>
        </fsm:top>
        <fsm:comment>Uses a faster search algorithm for searching for contacts</fsm:comment>
      </fsm:StateMachine>
    </fsm:submachine>
  </fsm:SubmachineState>
</fsm:subvertex>
<fsm:subvertex>
  <fsm:SubmachineState rdf:ID="EmergencyContactSubSM">
    <fsm:submachine>
      <fsm:StateMachine rdf:ID="EmergencyContact">
        <fsm:top>
          <fsm:CompositeState rdf:ID="EmergencyContactCS">
            <fsm:subvertex>
              <fsm:PseudoState rdf:ID="EmergencyContactInitialState">
                <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
                <fsm:outgoing>
                  <fsm:Transition rdf:ID="EmergencyContactInitialStateInitialTransition">
                    <fsm:trigger rdf:resource="#InitialEvent"/>
                    <fsm:source rdf:resource="#EmergencyContactInitialState"/>
                    <fsm:target>
                      <fsm:SimpleState rdf:ID="EmergencyContactState1">
                        <fsm:doActivity>useEmergencyContact()</fsm:doActivity>
                        <fsm:incoming rdf:resource="#EmergencyContactInitialStateInitialTransition"/>
                      </fsm:SimpleState>
                    </fsm:target>
                  </fsm:Transition>
                </fsm:outgoing>
              </fsm:PseudoState>
            </fsm:subvertex>
            <fsm:subvertex rdf:resource="#EmergencyContactState1"/>
          </fsm:CompositeState>
        </fsm:top>
        <fsm:comment>Provide a different contact for emergency cases</fsm:comment>
      </fsm:StateMachine>
    </fsm:submachine>
  </fsm:SubmachineState>
</fsm:subvertex>
<fsm:subvertex>
  <fsm:SubmachineState rdf:ID="TenaciousSubSM">
    <fsm:submachine>
      <fsm:StateMachine rdf:ID="Tenacious">
        <fsm:top>
          <fsm:CompositeState rdf:ID="TenaciousCS">
            <fsm:subvertex>
              <fsm:PseudoState rdf:ID="TenaciousInitialState">
                <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
                <fsm:outgoing>
                  <fsm:Transition rdf:ID="TenaciousInitialStateInitialTransition">
                    <fsm:trigger rdf:resource="#InitialEvent"/>
                    <fsm:source rdf:resource="#TenaciousInitialState"/>
                    <fsm:target>
                      <fsm:SimpleState rdf:ID="TenaciousState1">
                        <fsm:doActivity>setRetries(5)</fsm:doActivity>
                        <fsm:incoming rdf:resource="#TenaciousInitialStateInitialTransition"/>
                      </fsm:SimpleState>
                    </fsm:target>
                  </fsm:Transition>
                </fsm:outgoing>
              </fsm:PseudoState>
            </fsm:subvertex>
            <fsm:subvertex rdf:resource="#TenaciousState1"/>
          </fsm:CompositeState>
        </fsm:top>
        <fsm:comment>If the call is not answered, it tries again </fsm:comment>
      </fsm:StateMachine>
    </fsm:submachine>
  </fsm:SubmachineState>
</fsm:subvertex>
<fsm:subvertex rdf:resource="#RichContentSubSM"/>
<fsm:subvertex>
  <fsm:SubmachineState rdf:ID="EncryptionSubSM">
    <fsm:submachine>
      <fsm:StateMachine rdf:ID="Encryption">
        <fsm:top>
          <fsm:CompositeState rdf:ID="EncryptionCS">
            <fsm:subvertex>
              <fsm:PseudoState rdf:ID="EncryptionInitialState">
                <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
                <fsm:outgoing>
                  <fsm:Transition rdf:ID="EncryptionInitialStateInitialTransition">
                    <fsm:trigger rdf:resource="#InitialEvent"/>
                    <fsm:source rdf:resource="#EncryptionInitialState"/>
                    <fsm:target>
                      <fsm:FinalState rdf:ID="EncryptionState1">
                        <fsm:doActivity>setEncryption()</fsm:doActivity>
                        <fsm:incoming rdf:resource="#EncryptionInitialStateInitialTransition"/>
                      </fsm:FinalState>
                    </fsm:target>
                  </fsm:Transition>
                </fsm:outgoing>
              </fsm:PseudoState>
            </fsm:subvertex>
            <fsm:subvertex rdf:resource="#EncryptionState1"/>
          </fsm:CompositeState>
        </fsm:top>
        <fsm:comment>Encrypts the email message content</fsm:comment>
      </fsm:StateMachine>
    </fsm:submachine>
  </fsm:SubmachineState>
</fsm:subvertex>
<fsm:subvertex>
  <fsm:SubmachineState rdf:ID="AuthenticationSubSM">
    <fsm:submachine>
      <fsm:StateMachine rdf:ID="Authentication">
        <fsm:top>
          <fsm:CompositeState rdf:ID="AuthenticationCS">
            <fsm:subvertex rdf:resource="#AuthenticationInitialState"/>
            <fsm:subvertex rdf:resource="#AuthenticationState1"/>
          </fsm:CompositeState>
        </fsm:top>
        <fsm:comment>Authenticates the recipient by requesting a pin number before
notifying</fsm:comment>
      </fsm:StateMachine>
  </fsm:submachine>
 </fsm:SubmachineState>
</fsm:subvertex>
<fsm:incoming>
 <fsm:Transition rdf:ID="InputToProcess">
  <fsm:trigger>
   <fsm:SignalEvent rdf:ID="ProcessEvent">
    <fsm:signal>
     <fsm:Signal rdf:ID="ProcessEventSource">
      <fsm:value>ProcessSignal</fsm:value>
     </fsm:Signal>
    </fsm:signal>
   </fsm:SignalEvent>
  </fsm:trigger>
  <fsm:source>
   <fsm:CompositeState rdf:ID="InputState">
    <fsm:incoming>
     <fsm:Transition rdf:ID="IdleToInput">
      <fsm:trigger>
       <fsm:SignalEvent rdf:ID="InputEvent">
        <fsm:signal>
         <fsm:Signal rdf:ID="InputEventSource">
          <fsm:value>InputSignal</fsm:value>
         </fsm:Signal>
        </fsm:signal>
       </fsm:SignalEvent>
      </fsm:trigger>
      <fsm:source>
       <fsm:CompositeState rdf:ID="IdleState">
        <fsm:incoming>
         <fsm:Transition rdf:ID="InitialT">
          <fsm:trigger rdf:resource="#InitialEvent"/>
          <fsm:source>
           <fsm:PseudoState rdf:ID="InitialState">
            <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
            <fsm:outgoing rdf:resource="#InitialT"/>
           </fsm:PseudoState>
          </fsm:source>
          <fsm:target rdf:resource="#IdleState"/>
         </fsm:Transition>
        </fsm:incoming>
        <fsm:outgoing rdf:resource="#IdleToInput"/>
        <fsm:incoming>
         <fsm:Transition rdf:ID="OutputToIdle">
          <fsm:trigger>
           <fsm:SignalEvent rdf:ID="IdleEvent">
            <fsm:signal>
             <fsm:Signal rdf:ID="IdleEventSource">
              <fsm:value>IdleSignal</fsm:value>
             </fsm:Signal>
            </fsm:signal>
           </fsm:SignalEvent>
          </fsm:trigger>
          <fsm:source>
           <fsm:CompositeState rdf:ID="OutputState">
            <fsm:incoming>
             <fsm:Transition rdf:ID="ProcessToOutput">
              <fsm:trigger>
               <fsm:SignalEvent rdf:ID="OutputEvent">
                <fsm:signal>
```

```
                 <fsm:Signal rdf:ID="OutputEventSource">
                  <fsm:value>OutputSignal</fsm:value>
                 </fsm:Signal>
                </fsm:signal>
               </fsm:SignalEvent>
              </fsm:trigger>
              <fsm:source rdf:resource="#ProcessState"/>
              <fsm:target rdf:resource="#OutputState"/>
             </fsm:Transition>
            </fsm:incoming>
            <fsm:outgoing rdf:resource="#OutputToIdle"/>
           </fsm:CompositeState>
          </fsm:source>
          <fsm:target rdf:resource="#IdleState"/>
         </fsm:Transition>
        </fsm:incoming>
       </fsm:CompositeState>
      </fsm:source>
      <fsm:target rdf:resource="#InputState"/>
     </fsm:Transition>
    </fsm:incoming>
    <fsm:outgoing rdf:resource="#InputToProcess"/>
   </fsm:CompositeState>
  </fsm:source>
  <fsm:target rdf:resource="#ProcessState"/>
 </fsm:Transition>
</fsm:incoming>
<fsm:subvertex>
 <fsm:SubmachineState rdf:ID="MessageInFrenchSubSM">
  <fsm:submachine>
   <fsm:StateMachine rdf:ID="MessageInFrench">
    <fsm:top>
     <fsm:CompositeState rdf:ID="MessageInFrenchCS">
      <fsm:subvertex>
       <fsm:PseudoState rdf:ID="MessageInFrenchInitialState">
        <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
        <fsm:outgoing>
         <fsm:Transition rdf:ID="MessageInFrenchInitialStateInitialTransition">
          <fsm:trigger rdf:resource="#InitialEvent"/>
          <fsm:source rdf:resource="#MessageInFrenchInitialState"/>
          <fsm:target>
           <fsm:FinalState rdf:ID="MessageInFrenchState1">
            <fsm:doActivity>setLanguage(French)</fsm:doActivity>
            <fsm:incoming rdf:resource="#MessageInFrenchInitialStateInitialTransition"/>
           </fsm:FinalState>
          </fsm:target>
         </fsm:Transition>
        </fsm:outgoing>
       </fsm:PseudoState>
      </fsm:subvertex>
      <fsm:subvertex rdf:resource="#MessageInFrenchState1"/>
     </fsm:CompositeState>
    </fsm:top>
    <fsm:comment>Translates message to French</fsm:comment>
   </fsm:StateMachine>
  </fsm:submachine>
 </fsm:SubmachineState>
</fsm:subvertex>
<fsm:subvertex>
 <fsm:SubmachineState rdf:ID="SimpleSearchSubSM">
```

```
    <fsm:submachine>
      <fsm:StateMachine rdf:ID="SimpleSearch">
        <fsm:top>
          <fsm:CompositeState rdf:ID="SimpleSearchCS">
            <fsm:subvertex>
              <fsm:PseudoState rdf:ID="SimpleSearchInitialState">
                <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
                <fsm:outgoing>
                  <fsm:Transition rdf:ID="SimpleSearchInitialStateInitialTransition">
                    <fsm:trigger rdf:resource="#InitialEvent"/>
                    <fsm:source rdf:resource="#SimpleSearchInitialState"/>
                    <fsm:target>
                      <fsm:SimpleState rdf:ID="SimpleSearchState1">
                        <fsm:doActivity>doSimpleSort()</fsm:doActivity>
                        <fsm:incoming rdf:resource="#SimpleSearchInitialStateInitialTransition"/>
                        <fsm:outgoing>
                          <fsm:Transition rdf:ID="SimpleSearchState1Transition1">
                            <fsm:trigger rdf:resource="#SortedEvent"/>
                            <fsm:source rdf:resource="#SimpleSearchState1"/>
                            <fsm:target>
                              <fsm:FinalState rdf:ID="SimpleSearchState2">
                                <fsm:doActivity>doBinarySearch()</fsm:doActivity>
                                <fsm:incoming rdf:resource="#SimpleSearchState1Transition1"/>
                              </fsm:FinalState>
                            </fsm:target>
                          </fsm:Transition>
                        </fsm:outgoing>
                      </fsm:SimpleState>
                    </fsm:target>
                  </fsm:Transition>
                </fsm:outgoing>
              </fsm:PseudoState>
            </fsm:subvertex>
            <fsm:subvertex rdf:resource="#SimpleSearchState1"/>
            <fsm:subvertex rdf:resource="#SimpleSearchState2"/>
          </fsm:CompositeState>
        </fsm:top>
        <fsm:comment>Use simple sort and binary search</fsm:comment>
      </fsm:StateMachine>
    </fsm:submachine>
  </fsm:SubmachineState>
</fsm:subvertex>
<fsm:subvertex>
  <fsm:SubmachineState rdf:ID="ShortMessageModeSubSM">
    <fsm:submachine>
      <fsm:StateMachine rdf:ID="ShortMessageMode">
        <fsm:top>
          <fsm:CompositeState rdf:ID="ShortMessageModeCS">
            <fsm:subvertex>
              <fsm:PseudoState rdf:ID="ShortMessageModeInitialState">
                <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
                <fsm:outgoing>
                  <fsm:Transition rdf:ID="ShortMessageModeInitialStateInitialTransition">
                    <fsm:trigger rdf:resource="#InitialEvent"/>
                    <fsm:source rdf:resource="#ShortMessageModeInitialState"/>
                    <fsm:target>
                      <fsm:SimpleState rdf:ID="ShortMessageModeState1">
                        <fsm:doActivity>setShortMode()</fsm:doActivity>
                        <fsm:incoming rdf:resource="#ShortMessageModeInitialStateInitialTransition"/>
                      </fsm:SimpleState>
          </fsm:target>
         </fsm:Transition>
        </fsm:outgoing>
       </fsm:PseudoState>
      </fsm:subvertex>
      <fsm:subvertex rdf:resource="#ShortMessageModeState1"/>
     </fsm:CompositeState>
    </fsm:top>
    <fsm:comment></fsm:comment>
   </fsm:StateMachine>
  </fsm:submachine>
 </fsm:SubmachineState>
</fsm:subvertex>
<fsm:subvertex>
 <fsm:SubmachineState rdf:ID="CompressionSubSM">
  <fsm:submachine>
   <fsm:StateMachine rdf:ID="Compression">
    <fsm:top>
     <fsm:CompositeState rdf:ID="CompressionCS">
      <fsm:subvertex>
       <fsm:PseudoState rdf:ID="CompressionInitialState">
        <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
        <fsm:outgoing>
         <fsm:Transition rdf:ID="CompressionInitialStateInitialTransition">
          <fsm:trigger rdf:resource="#InitialEvent"/>
          <fsm:source rdf:resource="#CompressionInitialState"/>
          <fsm:target>
           <fsm:FinalState rdf:ID="CompressionState1">
            <fsm:doActivity>setCompression()</fsm:doActivity>
            <fsm:incoming rdf:resource="#CompressionInitialStateInitialTransition"/>
           </fsm:FinalState>
          </fsm:target>
         </fsm:Transition>
        </fsm:outgoing>
       </fsm:PseudoState>
      </fsm:subvertex>
      <fsm:subvertex rdf:resource="#CompressionState1"/>
     </fsm:CompositeState>
    </fsm:top>
    <fsm:comment>Compress email message content</fsm:comment>
   </fsm:StateMachine>
  </fsm:submachine>
 </fsm:SubmachineState>
</fsm:subvertex>
<fsm:subvertex>
 <fsm:SubmachineState rdf:ID="MessageInSpanishSubSM">
  <fsm:submachine>
   <fsm:StateMachine rdf:ID="MessageInSpanish">
    <fsm:top>
     <fsm:CompositeState rdf:ID="MessageInSpanishCS">
      <fsm:subvertex>
       <fsm:PseudoState rdf:ID="MessageInSpanishInitialState">
        <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
        <fsm:outgoing>
         <fsm:Transition rdf:ID="MessageInSpanishInitialStateInitialTransition">
          <fsm:trigger rdf:resource="#InitialEvent"/>
          <fsm:source rdf:resource="#MessageInSpanishInitialState"/>
          <fsm:target>
           <fsm:FinalState rdf:ID="MessageInSpanishState1">
            <fsm:doActivity>setLanguage(Spanish)</fsm:doActivity>
```

```
                            <fsm:incoming rdf:resource="#MessageInSpanishInitialStateInitialTransition"/>
                          </fsm:FinalState>
                        </fsm:target>
                      </fsm:Transition>
                    </fsm:outgoing>
                  </fsm:PseudoState>
                </fsm:subvertex>
                <fsm:subvertex rdf:resource="#MessageInSpanishState1"/>
              </fsm:CompositeState>
            </fsm:top>
            <fsm:comment>Translates message to Spanish</fsm:comment>
          </fsm:StateMachine>
        </fsm:submachine>
      </fsm:SubmachineState>
    </fsm:subvertex>
    <fsm:outgoing rdf:resource="#ProcessToOutput"/>
  </fsm:CompositeState>
  <fsm:CompositeState rdf:ID="ServiceState">
    <fsm:subvertex rdf:resource="#InitialState"/>
    <fsm:subvertex rdf:resource="#IdleState"/>
    <fsm:subvertex rdf:resource="#ProcessState"/>
    <fsm:subvertex rdf:resource="#InputState"/>
    <fsm:subvertex rdf:resource="#OutputState"/>
  </fsm:CompositeState>
  <fsm:StateMachine rdf:ID="NotificationServiceFSM">
    <fsm:top rdf:resource="#ServiceState"/>
  </fsm:StateMachine>
</rdf:RDF>
```

Notification Service's Management Policy

```
<rdf:RDF
    xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
    xmlns:policy="http://kdeg.cs.tcd.ie/Policy.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:thisprocess="http://www.daml.org/services/owl-s/1.1/NotificationProcess.owl#"
    xmlns:fsm="http://kdeg.cs.tcd.ie/FiniteStateMachine.owl#"
    xmlns="http://www.daml.org/services/owl-s/1.1/NotificationServicePolicy.owl#"
    xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
    xmlns:grounding="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
    xmlns:base="http://www.daml.org/services/owl-s/1.1/NotificationServicePolicy.owl"
    xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
  xml:base="http://www.daml.org/services/owl-s/1.1/NotificationServicePolicy.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://kdeg.cs.tcd.ie/Policy.owl"/>
    <owl:imports rdf:resource="NotificationProcess.owl"/>
  </owl:Ontology>
  <policy:SimpleEvent rdf:ID="NotificationServicePolicy2Event1">
    <policy:value>ProcessEvent</policy:value>
  </policy:SimpleEvent>
  <policy:SimpleAction rdf:ID="NotificationServicePolicy4Action1">
    <policy:value>MessageInFrench</policy:value>
  </policy:SimpleAction>
  <policy:SimpleAction rdf:ID="NotificationServicePolicy2Action102">
    <policy:value>ShortMessageMode</policy:value>
  </policy:SimpleAction>
```

```
<policy:SimpleCondition rdf:ID="NotificationServicePolicy2Condition1">
  <policy:subject>
    <policy:Subject rdf:ID="Membership"/>
  </policy:subject>
  <policy:predicate>
    <policy:Predicate rdf:ID="equal"/>
  </policy:predicate>
  <policy:value>Bronze</policy:value>
</policy:SimpleCondition>
<policy:Policy rdf:ID="NotificationServicePolicy1">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#NotificationService"/>
  <policy:event>
    <policy:SimpleEvent rdf:ID="NotificationServicePolicy1Event1">
      <policy:value>ProcessEvent</policy:value>
    </policy:SimpleEvent>
  </policy:event>
  <policy:condition>
    <policy:SimpleCondition rdf:ID="NotificationServicePolicy1Condition1">
      <policy:subject rdf:resource="#Membership"/>
      <policy:predicate rdf:resource="#equal"/>
      <policy:value>Gold</policy:value>
    </policy:SimpleCondition>
  </policy:condition>
  <policy:action>
    <policy:ComplexAction rdf:ID="NotificationServicePolicy1Action1">
      <rdfs:first>
        <policy:AndList>
          <rdfs:first>
            <policy:SimpleAction rdf:ID="NotificationServicePolicy1Action101">
              <policy:value>QuickSearch</policy:value>
            </policy:SimpleAction>
          </rdfs:first>
          <rdfs:rest>
            <policy:SimpleAction rdf:ID="NotificationServicePolicy1Action102">
              <policy:value>RichContent</policy:value>
            </policy:SimpleAction>
          </rdfs:rest>
        </policy:AndList>
      </rdfs:first>
      <rdfs:rest></rdfs:rest>
    </policy:ComplexAction>
  </policy:action>
</policy:Policy>
<policy:SimpleCondition rdf:ID="NotificationServicePolicy4Condition1">
  <policy:subject rdf:resource="#Membership"/>
  <policy:predicate rdf:resource="#equal"/>
  <policy:value>LeonClub</policy:value>
</policy:SimpleCondition>
<policy:Policy rdf:ID="NotificationServicePolicy4">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#NotificationService"/>
  <policy:event>
    <policy:SimpleEvent rdf:ID="NotificationServicePolicy4Event1">
      <policy:value>ProcessEvent</policy:value>
    </policy:SimpleEvent>
  </policy:event>
  <policy:condition rdf:resource="#NotificationServicePolicy4Condition1"/>
  <policy:action rdf:resource="#NotificationServicePolicy4Action1"/>
</policy:Policy>
```

```
<policy:Policy rdf:ID="NotificationServicePolicy3">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#NotificationService"/>
  <policy:event>
    <policy:SimpleEvent rdf:ID="NotificationServicePolicy3Event1">
      <policy:value>ProcessEvent</policy:value>
    </policy:SimpleEvent>
  </policy:event>
  <policy:condition>
    <policy:SimpleCondition rdf:ID="NotificationServicePolicy3Condition1">
      <policy:subject>
        <policy:Subject rdf:ID="Priority"/>
      </policy:subject>
      <policy:predicate rdf:resource="#equal"/>
      <policy:value>high</policy:value>
    </policy:SimpleCondition>
  </policy:condition>
  <policy:action>
    <policy:ComplexAction rdf:ID="NotificationServicePolicy3Action1">
      <rdfs:first>
        <policy:AndList>
          <rdfs:first>
            <policy:SimpleAction rdf:ID="NotificationServicePolicy3Action101">
              <policy:value>EmergencyContact</policy:value>
            </policy:SimpleAction>
          </rdfs:first>
          <rdfs:rest>
            <policy:SimpleAction rdf:ID="NotificationServicePolicy3Action102">
              <policy:value>Authentication</policy:value>
            </policy:SimpleAction>
          </rdfs:rest>
        </policy:AndList>
      </rdfs:first>
      <rdfs:rest></rdfs:rest>
    </policy:ComplexAction>
  </policy:action>
</policy:Policy>
<policy:SimpleAction rdf:ID="NotificationServicePolicy2Action101">
  <policy:value>SimpleSearch</policy:value>
</policy:SimpleAction>
<policy:Policy rdf:ID="NotificationServicePolicy2">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#NotificationService"/>
  <policy:event rdf:resource="#NotificationServicePolicy2Event1"/>
  <policy:condition rdf:resource="#NotificationServicePolicy2Condition1"/>
  <policy:action>
    <policy:ComplexAction rdf:ID="NotificationServicePolicy2Action1">
      <rdfs:first>
        <policy:AndList>
          <rdfs:first rdf:resource="#NotificationServicePolicy2Action101"/>
          <rdfs:rest rdf:resource="#NotificationServicePolicy2Action102"/>
        </policy:AndList>
      </rdfs:first>
      <rdfs:rest></rdfs:rest>
    </policy:ComplexAction>
  </policy:action>
</policy:Policy>
</rdf:RDF>
```

Notification Service's Refined Policies

```
<rdf:RDF
  xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
  xmlns:policy="http://kdeg.cs.tcd.ie/Policy.owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:thisprocess="http://www.daml.org/services/owl-s/1.1/NotificationProcess.owl#"
  xmlns:fsm="http://kdeg.cs.tcd.ie/FiniteStateMachine.owl#"
  xmlns="http://www.daml.org/services/owl-s/1.1/NotificationServiceRefinedPolicy.owl#"
  xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
  xmlns:grounding="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
  xmlns:base="http://www.daml.org/services/owl-s/1.1/NotificationServiceRefinedPolicy.owl"
  xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
 xml:base="http://www.daml.org/services/owl-s/1.1/NotificationServiceRefinedPolicy.owl">
 <owl:Ontology rdf:about="">
  <owl:imports rdf:resource="http://kdeg.cs.tcd.ie/Policy.owl"/>
  <owl:imports rdf:resource="NotificationProcess.owl"/>
 </owl:Ontology>
 <policy:Policy rdf:ID="AddressBookServicePolicy1C">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#AddressBookService"/>
  <policy:event>
   <policy:SimpleEvent rdf:ID="AddressBookServicePolicy1CEvent1A">
    <policy:value>RNotificationServicePolicy1Event1A</policy:value>
   </policy:SimpleEvent>
  </policy:event>
  <policy:action>
   <policy:SimpleAction rdf:ID="AddressBookServicePolicy1CAction1">
    <policy:value>event(QuickSearchEvent)</policy:value>
   </policy:SimpleAction>
  </policy:action>
 </policy:Policy>
 <policy:SimpleAction rdf:ID="MessageServicePolicy2CAction1">
  <policy:value>event(ShortMessageModeEvent)</policy:value>
 </policy:SimpleAction>
 <policy:ComplexAction rdf:ID="SimpleSearchAction1">
  <rdfs:first>
   <policy:AndList>
    <rdfs:first>
     <policy:SimpleAction rdf:ID="SimpleSearchAction01">
      <policy:value>doSimpleSort()</policy:value>
     </policy:SimpleAction>
    </rdfs:first>
    <rdfs:rest>
     <policy:SimpleAction rdf:ID="SimpleSearchAction02">
      <policy:value>event(SimpleSearchState1Event)</policy:value>
     </policy:SimpleAction>
    </rdfs:rest>
   </policy:AndList>
  </rdfs:first>
  <rdfs:rest></rdfs:rest>
 </policy:ComplexAction>
 <policy:SimpleEvent rdf:ID="RNotificationServicePolicy2Event1">
  <policy:value>ProcessEvent</policy:value>
 </policy:SimpleEvent>
 <policy:Policy rdf:ID="MessageInFrenchPolicy1">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#MessageService"/>
  <policy:event>
   <policy:ComplexEvent rdf:ID="MessageInFrenchEvent1a">
    <rdfs:first>
     <policy:AndList>
      <rdfs:first>
       <policy:SimpleEvent rdf:ID="MessageInFrenchEvent1b">
        <policy:value>ProcessEvent</policy:value>
       </policy:SimpleEvent>
      </rdfs:first>
      <rdfs:rest>
       <policy:SimpleAction rdf:ID="MessageInFrenchEvent1">
        <policy:value>MessageInFrenchEvent</policy:value>
       </policy:SimpleAction>
      </rdfs:rest>
     </policy:AndList>
    </rdfs:first>
    <rdfs:rest></rdfs:rest>
   </policy:ComplexEvent>
  </policy:event>
  <policy:action>
   <policy:SimpleAction rdf:ID="MessageInFrenchAction1">
    <policy:value>setLanguage(French)</policy:value>
   </policy:SimpleAction>
  </policy:action>
 </policy:Policy>
 <policy:SimpleEvent rdf:ID="MessageServicePolicy2CEvent1A">
  <policy:value>RNotificationServicePolicy2Event1A</policy:value>
 </policy:SimpleEvent>
 <policy:SimpleAction rdf:ID="ShortMessageModeEvent1">
  <policy:value>ShortMessageModeEvent</policy:value>
 </policy:SimpleAction>
 <policy:Policy rdf:ID="MessageServicePolicy3">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#MessageService"/>
  <policy:event>
   <policy:SimpleEvent rdf:ID="MessageServicePolicy3Event1C">
    <policy:value>RNotificationServicePolicy3Event1C</policy:value>
   </policy:SimpleEvent>
  </policy:event>
  <policy:condition>
   <policy:SimpleCondition rdf:ID="MessageServicePolicy3Condition1">
    <policy:subject>
     <policy:Subject rdf:ID="priority"/>
    </policy:subject>
    <policy:predicate>
     <policy:Predicate rdf:ID="equal"/>
    </policy:predicate>
    <policy:value>high</policy:value>
   </policy:SimpleCondition>
  </policy:condition>
  <policy:action>
   <policy:SimpleAction rdf:ID="MessageServicePolicy3Action1CU1">
    <policy:value>event(RNotificationServicePolicy3Event1CU1)</policy:value>
   </policy:SimpleAction>
  </policy:action>
 </policy:Policy>
 <policy:SimpleAction rdf:ID="ShortMessageModeAction1">
  <policy:value>setShortMode()</policy:value>
 </policy:SimpleAction>
```

```
<policy:SimpleAction rdf:ID="MessageServicePolicy4CAction1">
  <policy:value>event(MessageInFrenchEvent)</policy:value>
</policy:SimpleAction>
<policy:SimpleEvent rdf:ID="ContactServicePolicy1CEvent1A">
  <policy:value>RNotificationServicePolicy1Event1A</policy:value>
</policy:SimpleEvent>
<policy:SimpleEvent rdf:ID="RNotificationServicePolicy1Event1">
  <policy:value>ProcessEvent</policy:value>
</policy:SimpleEvent>
<policy:ComplexEvent rdf:ID="ShortMessageModeEvent1a">
  <rdfs:first>
    <policy:AndList>
      <rdfs:first>
        <policy:SimpleEvent rdf:ID="ShortMessageModeEvent1b">
          <policy:value>ProcessEvent</policy:value>
        </policy:SimpleEvent>
      </rdfs:first>
      <rdfs:rest rdf:resource="#ShortMessageModeEvent1"/>
    </policy:AndList>
  </rdfs:first>
  <rdfs:rest></rdfs:rest>
</policy:ComplexEvent>
<policy:SimpleCondition rdf:ID="LoginServicePolicy2Condition1">
  <policy:subject>
    <policy:Subject rdf:ID="member"/>
  </policy:subject>
  <policy:predicate rdf:resource="#equal"/>
  <policy:value>Bronze</policy:value>
</policy:SimpleCondition>
<policy:Policy rdf:ID="AddressBookServicePolicy3C">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#AddressBookService"/>
  <policy:event>
    <policy:SimpleEvent rdf:ID="AddressBookServicePolicy3CEvent1A">
      <policy:value>RNotificationServicePolicy3Event1A</policy:value>
    </policy:SimpleEvent>
  </policy:event>
  <policy:action>
    <policy:SimpleAction rdf:ID="AddressBookServicePolicy3CAction1">
      <policy:value>event(EmergencyContactEvent)</policy:value>
    </policy:SimpleAction>
  </policy:action>
</policy:Policy>
<policy:SimpleEvent rdf:ID="LoginServicePolicy4Event1C">
  <policy:value>RNotificationServicePolicy4Event1C</policy:value>
</policy:SimpleEvent>
<policy:SimpleAction rdf:ID="AuthenticationAction1">
  <policy:value>setAuthentication()</policy:value>
</policy:SimpleAction>
<policy:SimpleEvent rdf:ID="SimpleSearchEvent001">
  <policy:value>SortedEvent</policy:value>
</policy:SimpleEvent>
<policy:SimpleEvent rdf:ID="RNotificationServicePolicy4Event1">
  <policy:value>ProcessEvent</policy:value>
</policy:SimpleEvent>
<policy:ComplexEvent rdf:ID="RichContentEvent2">
  <rdfs:first>
    <policy:AndList>
      <rdfs:first>
        <policy:SimpleEvent rdf:ID="RichContentEvent001">
```

```
          <policy:value>MessageCreatedEvent</policy:value>
        </policy:SimpleEvent>
      </rdfs:first>
      <rdfs:rest>
        <policy:SimpleAction rdf:ID="RichContentEvent002">
          <policy:value>RichContentState1Event</policy:value>
        </policy:SimpleAction>
      </rdfs:rest>
    </policy:AndList>
  </rdfs:first>
  <rdfs:rest></rdfs:rest>
</policy:ComplexEvent>
<policy:ComplexEvent rdf:ID="AuthenticationEvent1a">
  <rdfs:first>
    <policy:AndList>
      <rdfs:first>
        <policy:SimpleEvent rdf:ID="AuthenticationEvent1b">
          <policy:value>ProcessEvent</policy:value>
        </policy:SimpleEvent>
      </rdfs:first>
      <rdfs:rest>
        <policy:SimpleAction rdf:ID="AuthenticationEvent1">
          <policy:value>AuthenticationEvent</policy:value>
        </policy:SimpleAction>
      </rdfs:rest>
    </policy:AndList>
  </rdfs:first>
  <rdfs:rest></rdfs:rest>
</policy:ComplexEvent>
<policy:SimpleEvent rdf:ID="RNotificationServicePolicy3Event1">
  <policy:value>ProcessEvent</policy:value>
</policy:SimpleEvent>
<policy:Policy rdf:ID="RNotificationServicePolicy4">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#NotificationService"/>
  <policy:event rdf:resource="#RNotificationServicePolicy4Event1"/>
  <policy:action>
    <policy:SimpleAction rdf:ID="RNotificationServicePolicy4Action1C">
      <policy:value>event(RNotificationServicePolicy4Event1C)</policy:value>
    </policy:SimpleAction>
  </policy:action>
</policy:Policy>
<policy:Policy rdf:ID="AddressBookServicePolicy2C">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#AddressBookService"/>
  <policy:event>
    <policy:SimpleEvent rdf:ID="AddressBookServicePolicy2CEvent1A">
      <policy:value>RNotificationServicePolicy2Event1A</policy:value>
    </policy:SimpleEvent>
  </policy:event>
  <policy:action>
    <policy:SimpleAction rdf:ID="AddressBookServicePolicy2CAction1">
      <policy:value>event(SimpleSearchEvent)</policy:value>
    </policy:SimpleAction>
  </policy:action>
</policy:Policy>
<policy:SimpleAction rdf:ID="SimpleSearchEvent002">
  <policy:value>SimpleSearchState1Event</policy:value>
</policy:SimpleAction>
<policy:SimpleAction rdf:ID="LoginServicePolicy2Action1CU0">
```

```
  <policy:value>event(RNotificationServicePolicy2Event1CU0)</policy:value>
</policy:SimpleAction>
<policy:ComplexEvent rdf:ID="SimpleSearchEvent2">
  <rdfs:first>
    <policy:AndList>
      <rdfs:first rdf:resource="#SimpleSearchEvent001"/>
      <rdfs:rest rdf:resource="#SimpleSearchEvent002"/>
    </policy:AndList>
  </rdfs:first>
  <rdfs:rest></rdfs:rest>
</policy:ComplexEvent>
<policy:Policy rdf:ID="RNotificationServicePolicy2CU">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#NotificationService"/>
  <policy:event>
    <policy:SimpleEvent rdf:ID="LoginServicePolicy2Event1CU">
      <policy:value>RNotificationServicePolicy2Event1CU0</policy:value>
    </policy:SimpleEvent>
  </policy:event>
  <policy:action>
    <policy:SimpleAction rdf:ID="RNotificationServicePolicy2Action1CU">
      <policy:value>event(RNotificationServicePolicy2Event1A)</policy:value>
    </policy:SimpleAction>
  </policy:action>
</policy:Policy>
<policy:Policy rdf:ID="LoginServicePolicy4">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#LoginService"/>
  <policy:event rdf:resource="#LoginServicePolicy4Event1C"/>
  <policy:condition>
    <policy:SimpleCondition rdf:ID="LoginServicePolicy4Condition1">
      <policy:subject rdf:resource="#member"/>
      <policy:predicate rdf:resource="#equal"/>
      <policy:value>LeonClub</policy:value>
    </policy:SimpleCondition>
  </policy:condition>
  <policy:action>
    <policy:SimpleAction rdf:ID="LoginServicePolicy4Action1CU0">
      <policy:value>event(RNotificationServicePolicy4Event1CU0)</policy:value>
    </policy:SimpleAction>
  </policy:action>
</policy:Policy>
<policy:ComplexEvent rdf:ID="EmergencyContactEvent1a">
  <rdfs:first>
    <policy:AndList>
      <rdfs:first>
        <policy:SimpleEvent rdf:ID="EmergencyContactEvent1b">
          <policy:value>ProcessEvent</policy:value>
        </policy:SimpleEvent>
      </rdfs:first>
      <rdfs:rest>
        <policy:SimpleAction rdf:ID="EmergencyContactEvent1">
          <policy:value>EmergencyContactEvent</policy:value>
        </policy:SimpleAction>
      </rdfs:rest>
    </policy:AndList>
  </rdfs:first>
  <rdfs:rest></rdfs:rest>
</policy:ComplexEvent>
<policy:Policy rdf:ID="RNotificationServicePolicy4CU">
```

```
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#NotificationService"/>
  <policy:event>
   <policy:SimpleEvent rdf:ID="LoginServicePolicy4Event1CU">
     <policy:value>RNotificationServicePolicy4Event1CU0</policy:value>
   </policy:SimpleEvent>
  </policy:event>
  <policy:action>
   <policy:SimpleAction rdf:ID="RNotificationServicePolicy4Action1CU">
     <policy:value>event(RNotificationServicePolicy4Event1A)</policy:value>
   </policy:SimpleAction>
  </policy:action>
 </policy:Policy>
 <policy:Policy rdf:ID="RNotificationServicePolicy1">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#NotificationService"/>
  <policy:event rdf:resource="#RNotificationServicePolicy1Event1"/>
  <policy:action>
   <policy:SimpleAction rdf:ID="RNotificationServicePolicy1Action1C">
     <policy:value>event(RNotificationServicePolicy1Event1C)</policy:value>
   </policy:SimpleAction>
  </policy:action>
 </policy:Policy>
 <policy:ComplexEvent rdf:ID="QuickSearchEvent2">
  <rdfs:first>
   <policy:AndList>
    <rdfs:first>
     <policy:SimpleEvent rdf:ID="QuickSearchEvent001">
      <policy:value>SortedEvent</policy:value>
     </policy:SimpleEvent>
    </rdfs:first>
    <rdfs:rest>
     <policy:SimpleAction rdf:ID="QuickSearchEvent002">
      <policy:value>QuickSearchState1Event</policy:value>
     </policy:SimpleAction>
    </rdfs:rest>
   </policy:AndList>
  </rdfs:first>
  <rdfs:rest></rdfs:rest>
 </policy:ComplexEvent>
 <policy:Policy rdf:ID="RichContentPolicy1">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#EmailService"/>
  <policy:event>
   <policy:ComplexEvent rdf:ID="RichContentEvent1a">
    <rdfs:first>
     <policy:AndList>
      <rdfs:first>
       <policy:SimpleEvent rdf:ID="RichContentEvent1b">
        <policy:value>ProcessEvent</policy:value>
       </policy:SimpleEvent>
      </rdfs:first>
      <rdfs:rest>
       <policy:SimpleAction rdf:ID="RichContentEvent1">
        <policy:value>RichContentEvent</policy:value>
       </policy:SimpleAction>
      </rdfs:rest>
     </policy:AndList>
    </rdfs:first>
    <rdfs:rest></rdfs:rest>
    </policy:ComplexEvent>
  </policy:event>
  <policy:action>
    <policy:ComplexAction rdf:ID="RichContentAction1">
      <rdfs:first>
        <policy:AndList>
          <rdfs:first>
            <policy:SimpleAction rdf:ID="RichContentAction01">
              <policy:value>doHtmlMessage()</policy:value>
            </policy:SimpleAction>
          </rdfs:first>
          <rdfs:rest>
            <policy:SimpleAction rdf:ID="RichContentAction02">
              <policy:value>event(RichContentState1Event)</policy:value>
            </policy:SimpleAction>
          </rdfs:rest>
        </policy:AndList>
      </rdfs:first>
      <rdfs:rest></rdfs:rest>
    </policy:ComplexAction>
  </policy:action>
</policy:Policy>
<policy:Policy rdf:ID="SimpleSearchPolicy2">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#AddressBookService"/>
  <policy:event rdf:resource="#SimpleSearchEvent2"/>
  <policy:action>
    <policy:SimpleAction rdf:ID="SimpleSearchAction2">
      <policy:value>doBinarySearch()</policy:value>
    </policy:SimpleAction>
  </policy:action>
</policy:Policy>
<policy:Policy rdf:ID="AuthenticationPolicy1">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#PhoneService"/>
  <policy:event rdf:resource="#AuthenticationEvent1a"/>
  <policy:action rdf:resource="#AuthenticationAction1"/>
</policy:Policy>
<policy:SimpleEvent rdf:ID="MessageServicePolicy4CEvent1A">
  <policy:value>RNotificationServicePolicy4Event1A</policy:value>
</policy:SimpleEvent>
<policy:Policy rdf:ID="RNotificationServicePolicy3">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#NotificationService"/>
  <policy:event rdf:resource="#RNotificationServicePolicy3Event1"/>
  <policy:action>
    <policy:SimpleAction rdf:ID="RNotificationServicePolicy3Action1C">
      <policy:value>event(RNotificationServicePolicy3Event1C)</policy:value>
    </policy:SimpleAction>
  </policy:action>
</policy:Policy>
<policy:SimpleAction rdf:ID="RNotificationServicePolicy2Action1C">
  <policy:value>event(RNotificationServicePolicy2Event1C)</policy:value>
</policy:SimpleAction>
<policy:Policy rdf:ID="EmergencyContactPolicy1">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#AddressBookService"/>
  <policy:event rdf:resource="#EmergencyContactEvent1a"/>
  <policy:action>
    <policy:SimpleAction rdf:ID="EmergencyContactAction1">
```

```
      <policy:value>useEmergencyContact()</policy:value>
    </policy:SimpleAction>
  </policy:action>
</policy:Policy>
<policy:SimpleEvent rdf:ID="LoginServicePolicy1Event1C">
  <policy:value>RNotificationServicePolicy1Event1C</policy:value>
</policy:SimpleEvent>
<policy:SimpleAction rdf:ID="QuickSearchEvent1">
  <policy:value>QuickSearchEvent</policy:value>
</policy:SimpleAction>
<policy:Policy rdf:ID="RichContentPolicy2">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#EmailService"/>
  <policy:event rdf:resource="#RichContentEvent2"/>
  <policy:action>
    <policy:SimpleAction rdf:ID="RichContentAction2">
      <policy:value>doHmtlHeader()</policy:value>
    </policy:SimpleAction>
  </policy:action>
</policy:Policy>
<policy:SimpleAction rdf:ID="ContactServicePolicy1CAction1">
  <policy:value>event(ContactServicePolicy1CEvent2A)</policy:value>
</policy:SimpleAction>
<policy:Policy rdf:ID="LoginServicePolicy1">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#LoginService"/>
  <policy:event rdf:resource="#LoginServicePolicy1Event1C"/>
  <policy:condition>
    <policy:SimpleCondition rdf:ID="LoginServicePolicy1Condition1">
      <policy:subject rdf:resource="#member"/>
      <policy:predicate rdf:resource="#equal"/>
      <policy:value>Gold</policy:value>
    </policy:SimpleCondition>
  </policy:condition>
  <policy:action>
    <policy:SimpleAction rdf:ID="LoginServicePolicy1Action1CU0">
      <policy:value>event(RNotificationServicePolicy1Event1CU0)</policy:value>
    </policy:SimpleAction>
  </policy:action>
</policy:Policy>
<policy:ComplexAction rdf:ID="QuickSearchAction1">
  <rdfs:first>
    <policy:AndList>
      <rdfs:first>
        <policy:SimpleAction rdf:ID="QuickSearchAction01">
          <policy:value>doQuickSort()</policy:value>
        </policy:SimpleAction>
      </rdfs:first>
      <rdfs:rest>
        <policy:SimpleAction rdf:ID="QuickSearchAction02">
          <policy:value>event(QuickSearchState1Event)</policy:value>
        </policy:SimpleAction>
      </rdfs:rest>
    </policy:AndList>
  </rdfs:first>
  <rdfs:rest></rdfs:rest>
</policy:ComplexAction>
<policy:SimpleEvent rdf:ID="EmailServicePolicy1CCEvent2A">
  <policy:value>ContactServicePolicy1CEvent2A</policy:value>
</policy:SimpleEvent>
```

```
<policy:SimpleEvent rdf:ID="PhoneServicePolicy3CCEvent2A">
  <policy:value>ContactServicePolicy3CEvent2A</policy:value>
</policy:SimpleEvent>
<policy:Policy rdf:ID="ContactServicePolicy3C">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#ContactService"/>
  <policy:event>
    <policy:SimpleEvent rdf:ID="ContactServicePolicy3CEvent1A">
      <policy:value>RNotificationServicePolicy3Event1A</policy:value>
    </policy:SimpleEvent>
  </policy:event>
  <policy:action>
    <policy:SimpleAction rdf:ID="ContactServicePolicy3CAction1">
      <policy:value>event(ContactServicePolicy3CEvent2A)</policy:value>
    </policy:SimpleAction>
  </policy:action>
</policy:Policy>
<policy:SimpleEvent rdf:ID="SimpleSearchEvent1b">
  <policy:value>ProcessEvent</policy:value>
</policy:SimpleEvent>
<policy:SimpleAction rdf:ID="EmailServicePolicy1CCAction2">
  <policy:value>event(RichContentEvent)</policy:value>
</policy:SimpleAction>
<policy:SimpleAction rdf:ID="QuickSearchAction2">
  <policy:value>doBinarySearch()</policy:value>
</policy:SimpleAction>
<policy:Policy rdf:ID="RNotificationServicePolicy2">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#NotificationService"/>
  <policy:event rdf:resource="#RNotificationServicePolicy2Event1"/>
  <policy:action rdf:resource="#RNotificationServicePolicy2Action1C"/>
</policy:Policy>
<policy:ComplexEvent rdf:ID="SimpleSearchEvent1a">
  <rdfs:first>
    <policy:AndList>
      <rdfs:first rdf:resource="#SimpleSearchEvent1b"/>
      <rdfs:rest>
        <policy:SimpleAction rdf:ID="SimpleSearchEvent1">
          <policy:value>SimpleSearchEvent</policy:value>
        </policy:SimpleAction>
      </rdfs:rest>
    </policy:AndList>
  </rdfs:first>
  <rdfs:rest></rdfs:rest>
</policy:ComplexEvent>
<policy:SimpleEvent rdf:ID="MessageServicePolicy3Event1CU">
  <policy:value>RNotificationServicePolicy3Event1CU1</policy:value>
</policy:SimpleEvent>
<policy:Policy rdf:ID="QuickSearchPolicy1">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#AddressBookService"/>
  <policy:event>
    <policy:ComplexEvent rdf:ID="QuickSearchEvent1a">
      <rdfs:first>
        <policy:AndList>
          <rdfs:first>
            <policy:SimpleEvent rdf:ID="QuickSearchEvent1b">
              <policy:value>ProcessEvent</policy:value>
            </policy:SimpleEvent>
          </rdfs:first>
      <rdfs:rest rdf:resource="#QuickSearchEvent1"/>
     </policy:AndList>
    </rdfs:first>
    <rdfs:rest></rdfs:rest>
   </policy:ComplexEvent>
  </policy:event>
  <policy:action rdf:resource="#QuickSearchAction1"/>
</policy:Policy>
<policy:SimpleAction rdf:ID="PhoneServicePolicy3CCAction2">
  <policy:value>event(AuthenticationEvent)</policy:value>
</policy:SimpleAction>
<policy:Policy rdf:ID="RNotificationServicePolicy3CU">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#NotificationService"/>
  <policy:event rdf:resource="#MessageServicePolicy3Event1CU"/>
  <policy:action>
   <policy:SimpleAction rdf:ID="RNotificationServicePolicy3Action1CU">
    <policy:value>event(RNotificationServicePolicy3Event1A)</policy:value>
   </policy:SimpleAction>
  </policy:action>
</policy:Policy>
<policy:Policy rdf:ID="QuickSearchPolicy2">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#AddressBookService"/>
  <policy:event rdf:resource="#QuickSearchEvent2"/>
  <policy:action rdf:resource="#QuickSearchAction2"/>
</policy:Policy>
<policy:Policy rdf:ID="ShortMessageModePolicy1">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#MessageService"/>
  <policy:event rdf:resource="#ShortMessageModeEvent1a"/>
  <policy:action rdf:resource="#ShortMessageModeAction1"/>
</policy:Policy>
<policy:Policy rdf:ID="MessageServicePolicy4C">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#MessageService"/>
  <policy:event rdf:resource="#MessageServicePolicy4CEvent1A"/>
  <policy:action rdf:resource="#MessageServicePolicy4CAction1"/>
</policy:Policy>
<policy:SimpleEvent rdf:ID="LoginServicePolicy1Event1CU">
  <policy:value>RNotificationServicePolicy1Event1CU0</policy:value>
</policy:SimpleEvent>
<policy:Policy rdf:ID="RNotificationServicePolicy1CU">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#NotificationService"/>
  <policy:event rdf:resource="#LoginServicePolicy1Event1CU"/>
  <policy:action>
   <policy:SimpleAction rdf:ID="RNotificationServicePolicy1Action1CU">
    <policy:value>event(RNotificationServicePolicy1Event1A)</policy:value>
   </policy:SimpleAction>
  </policy:action>
</policy:Policy>
<policy:Policy rdf:ID="ContactServicePolicy1C">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="NotificationProcess.owl#ContactService"/>
  <policy:event rdf:resource="#ContactServicePolicy1CEvent1A"/>
  <policy:action rdf:resource="#ContactServicePolicy1CAction1"/>
</policy:Policy>
<policy:Policy rdf:ID="EmailServicePolicy1CC">
  <rdfs:comment></rdfs:comment>
```

```
    <policy:target rdf:resource="NotificationProcess.owl#EmailService"/>
    <policy:event rdf:resource="#EmailServicePolicy1CCEvent2A"/>
    <policy:action rdf:resource="#EmailServicePolicy1CCAction2"/>
  </policy:Policy>
  <policy:Policy rdf:ID="MessageServicePolicy2C">
    <rdfs:comment></rdfs:comment>
    <policy:target rdf:resource="NotificationProcess.owl#MessageService"/>
    <policy:event rdf:resource="#MessageServicePolicy2CEvent1A"/>
    <policy:action rdf:resource="#MessageServicePolicy2CAction1"/>
  </policy:Policy>
  <policy:Policy rdf:ID="PhoneServicePolicy3CC">
    <rdfs:comment></rdfs:comment>
    <policy:target rdf:resource="NotificationProcess.owl#PhoneService"/>
    <policy:event rdf:resource="#PhoneServicePolicy3CCEvent2A"/>
    <policy:action rdf:resource="#PhoneServicePolicy3CCAction2"/>
  </policy:Policy>
  <policy:SimpleEvent rdf:ID="LoginServicePolicy2Event1C">
    <policy:value>RNotificationServicePolicy2Event1C</policy:value>
  </policy:SimpleEvent>
  <policy:Policy rdf:ID="SimpleSearchPolicy1">
    <rdfs:comment></rdfs:comment>
    <policy:target rdf:resource="NotificationProcess.owl#AddressBookService"/>
    <policy:event rdf:resource="#SimpleSearchEvent1a"/>
    <policy:action rdf:resource="#SimpleSearchAction1"/>
  </policy:Policy>
  <policy:Policy rdf:ID="LoginServicePolicy2">
    <rdfs:comment></rdfs:comment>
    <policy:target rdf:resource="NotificationProcess.owl#LoginService"/>
    <policy:event rdf:resource="#LoginServicePolicy2Event1C"/>
    <policy:condition rdf:resource="#LoginServicePolicy2Condition1"/>
    <policy:action rdf:resource="#LoginServicePolicy2Action1CU0"/>
  </policy:Policy>
</rdf:RDF>
```

Notification Service's Refined Policies as Jess Rules

```
(defrule MessageServicePolicy3
(event (service ?service0&MessageService|Any) (type ?type0) (name
?event0&RNotificationServicePolicy3Event1C)) (event (service ?cservice0) (type ?ctype0) (name
?cevent0&~IdleEvent))
(param (service ?cservice0&MessageService) (direction ?direction0) (name ?param0&priority) (value
?value0)) (test (eq ?value0 high))
 =>(assert (event (service Any) (type Policy) (name RNotificationServicePolicy3Event1CU1))) (assert
(action (service ?service0) (name ?event0))))
(defrule LoginServicePolicy2
(event (service ?service0&LoginService|Any) (type ?type0) (name
?event0&RNotificationServicePolicy2Event1C)) (event (service ?cservice0) (type ?ctype0) (name
?cevent0&~IdleEvent))
(param (service ?cservice0&LoginService) (direction ?direction0) (name ?param0&member) (value
?value0)) (test (eq ?value0 Bronze))
 =>(assert (event (service Any) (type Policy) (name RNotificationServicePolicy2Event1CU0))) (assert
(action (service ?service0) (name ?event0))))
(defrule MessageServicePolicy4C
(event (service ?service0&MessageService|Any) (type ?type0) (name
?event0&RNotificationServicePolicy4Event1A))
=>(assert (event (service Any) (type Policy) (name MessageInFrenchEvent))) (assert (action (service
?service0) (name ?event0))))
(defrule MessageInFrenchPolicy1
(and (event (service ?service0&MessageService|Any) (type ?type0) (name ?event0&ProcessEvent))
(event (service ?service1&MessageService|Any) (type ?type1) (name
?event1&MessageInFrenchEvent)))
=>(service-state MessageService "setLanguage(French)") (assert (action (service ?service1) (name
?event1))))
(defrule LoginServicePolicy1
(event (service ?service0&LoginService|Any) (type ?type0) (name
?event0&RNotificationServicePolicy1Event1C)) (event (service ?cservice0) (type ?ctype0) (name
?cevent0&~IdleEvent))
(param (service ?cservice0&LoginService) (direction ?direction0) (name ?param0&member) (value
?value0)) (test (eq ?value0 Gold))
 =>(assert (event (service Any) (type Policy) (name RNotificationServicePolicy1Event1CU0))) (assert
(action (service ?service0) (name ?event0))))
(defrule RNotificationServicePolicy1
(event (service ?service0&NotificationService|Any) (type ?type0) (name ?event0&ProcessEvent))
=>(assert (event (service Any) (type Policy) (name RNotificationServicePolicy1Event1C))))
(defrule RichContentPolicy2
(and (event (service ?service0&EmailService|Any) (type ?type0) (name
?event0&MessageCreatedEvent)) (event (service ?service1&EmailService|Any) (type ?type1) (name
?event1&RichContentState1Event)))
=>(service-state EmailService "doHmtlHeader()") (assert (action (service ?service0) (name ?event0)))
(assert (action (service ?service1) (name ?event1))))
(defrule PhoneServicePolicy3CC
(event (service ?service0&PhoneService|Any) (type ?type0) (name
?event0&ContactServicePolicy3CEvent2A))
=>(assert (event (service Any) (type Policy) (name AuthenticationEvent))) (assert (action (service
?service0) (name ?event0))))
(defrule RNotificationServicePolicy4
(event (service ?service0&NotificationService|Any) (type ?type0) (name ?event0&ProcessEvent))
=>(assert (event (service Any) (type Policy) (name RNotificationServicePolicy4Event1C))))
(defrule ShortMessageModePolicy1
(and (event (service ?service0&MessageService|Any) (type ?type0) (name ?event0&ProcessEvent))
(event (service ?service1&MessageService|Any) (type ?type1) (name
?event1&ShortMessageModeEvent)))
=>(service-state MessageService "setShortMode()") (assert (action (service ?service1) (name
?event1))))
(defrule AddressBookServicePolicy1C
(event (service ?service0&AddressBookService|Any) (type ?type0) (name
?event0&RNotificationServicePolicy1Event1A))
=>(assert (event (service Any) (type Policy) (name QuickSearchEvent))) (assert (action (service
?service0) (name ?event0))))
(defrule AuthenticationPolicy1
(and (event (service ?service0&PhoneService|Any) (type ?type0) (name ?event0&ProcessEvent))
(event (service ?service1&PhoneService|Any) (type ?type1) (name ?event1&AuthenticationEvent)))
=>(service-state PhoneService "setAuthentication()") (assert (action (service ?service1) (name
?event1))))
(defrule AddressBookServicePolicy2C
(event (service ?service0&AddressBookService|Any) (type ?type0) (name
?event0&RNotificationServicePolicy2Event1A))
=>(assert (event (service Any) (type Policy) (name SimpleSearchEvent))) (assert (action (service
?service0) (name ?event0))))
(defrule RNotificationServicePolicy3CU
(event (service ?service0&NotificationService|Any) (type ?type0) (name
?event0&RNotificationServicePolicy3Event1CU1))
=>(assert (event (service Any) (type Policy) (name RNotificationServicePolicy3Event1A))) (assert
(action (service ?service0) (name ?event0))))
(defrule AddressBookServicePolicy3C
(event (service ?service0&AddressBookService|Any) (type ?type0) (name
?event0&RNotificationServicePolicy3Event1A))
```

```
=>(assert (event (service Any) (type Policy) (name EmergencyContactEvent))) (assert (action (service
?service0) (name ?event0))))
(defrule QuickSearchPolicy1
(and (event (service ?service0&AddressBookService|Any) (type ?type0) (name
?event0&ProcessEvent)) (event (service ?service1&AddressBookService|Any) (type ?type1) (name
?event1&QuickSearchEvent)))
=>(and (service-state AddressBookService "doQuickSort()") (assert (event (service Any) (type Policy)
(name QuickSearchState1Event)))) (assert (action (service ?service1) (name ?event1))))
(defrule EmergencyContactPolicy1
(and (event (service ?service0&AddressBookService|Any) (type ?type0) (name
?event0&ProcessEvent)) (event (service ?service1&AddressBookService|Any) (type ?type1) (name
?event1&EmergencyContactEvent)))
=>(service-state AddressBookService "useEmergencyContact()") (assert (action (service ?service1)
(name ?event1))))
(defrule RNotificationServicePolicy4CU
(event (service ?service0&NotificationService|Any) (type ?type0) (name
?event0&RNotificationServicePolicy4Event1CU0))
=>(assert (event (service Any) (type Policy) (name RNotificationServicePolicy4Event1A))) (assert
(action (service ?service0) (name ?event0))))
(defrule RichContentPolicy1
(and (event (service ?service0&EmailService|Any) (type ?type0) (name ?event0&ProcessEvent))
(event (service ?service1&EmailService|Any) (type ?type1) (name ?event1&RichContentEvent)))
=>(and (service-state EmailService "doHtmlMessage()") (assert (event (service Any) (type Policy)
(name RichContentState1Event)))) (assert (action (service ?service1) (name ?event1))))
(defrule ContactServicePolicy3C
(event (service ?service0&ContactService|Any) (type ?type0) (name
?event0&RNotificationServicePolicy3Event1A))
=>(assert (event (service Any) (type Policy) (name ContactServicePolicy3CEvent2A))) (assert (action
(service ?service0) (name ?event0))))
(defrule MessageServicePolicy2C
(event (service ?service0&MessageService|Any) (type ?type0) (name
?event0&RNotificationServicePolicy2Event1A))
=>(assert (event (service Any) (type Policy) (name ShortMessageModeEvent))) (assert (action (service
?service0) (name ?event0))))
(defrule RNotificationServicePolicy1CU
(event (service ?service0&NotificationService|Any) (type ?type0) (name
?event0&RNotificationServicePolicy1Event1CU0))
=>(assert (event (service Any) (type Policy) (name RNotificationServicePolicy1Event1A))) (assert
(action (service ?service0) (name ?event0))))
(defrule QuickSearchPolicy2
(and (event (service ?service0&AddressBookService|Any) (type ?type0) (name ?event0&SortedEvent))
(event (service ?service1&AddressBookService|Any) (type ?type1) (name
?event1&QuickSearchState1Event)))
=>(service-state AddressBookService "doBinarySearch()") (assert (action (service ?service0) (name
?event0))) (assert (action (service ?service1) (name ?event1))))
(defrule SimpleSearchPolicy1
(and (event (service ?service0&AddressBookService|Any) (type ?type0) (name
?event0&ProcessEvent)) (event (service ?service1&AddressBookService|Any) (type ?type1) (name
?event1&SimpleSearchEvent)))
=>(and (service-state AddressBookService "doSimpleSort()") (assert (event (service Any) (type Policy)
(name SimpleSearchState1Event)))) (assert (action (service ?service1) (name ?event1))))
(defrule EmailServicePolicy1CC
(event (service ?service0&EmailService|Any) (type ?type0) (name
?event0&ContactServicePolicy1CEvent2A))
=>(assert (event (service Any) (type Policy) (name RichContentEvent))) (assert (action (service
?service0) (name ?event0))))
(defrule SimpleSearchPolicy2
(and (event (service ?service0&AddressBookService|Any) (type ?type0) (name ?event0&SortedEvent))
(event (service ?service1&AddressBookService|Any) (type ?type1) (name
?event1&SimpleSearchState1Event)))
```

```
=>(service-state AddressBookService "doBinarySearch()") (assert (action (service ?service0) (name
?event0))) (assert (action (service ?service1) (name ?event1))))
(defrule LoginServicePolicy4
(event (service ?service0&LoginService|Any) (type ?type0) (name
?event0&RNotificationServicePolicy4Event1C)) (event (service ?cservice0) (type ?ctype0) (name
?cevent0&~IdleEvent))
(param (service ?cservice0&LoginService) (direction ?direction0) (name ?param0&member) (value
?value0)) (test (eq ?value0 LeonClub))
 =>(assert (event (service Any) (type Policy) (name RNotificationServicePolicy4Event1CU0))) (assert
(action (service ?service0) (name ?event0))))
(defrule ContactServicePolicy1C
(event (service ?service0&ContactService|Any) (type ?type0) (name
?event0&RNotificationServicePolicy1Event1A))
=>(assert (event (service Any) (type Policy) (name ContactServicePolicy1CEvent2A))) (assert (action
(service ?service0) (name ?event0))))
(defrule RNotificationServicePolicy2
(event (service ?service0&NotificationService|Any) (type ?type0) (name ?event0&ProcessEvent))
=>(assert (event (service Any) (type Policy) (name RNotificationServicePolicy2Event1C))))
(defrule RNotificationServicePolicy2CU
(event (service ?service0&NotificationService|Any) (type ?type0) (name
?event0&RNotificationServicePolicy2Event1CU0))
=>(assert (event (service Any) (type Policy) (name RNotificationServicePolicy2Event1A))) (assert
(action (service ?service0) (name ?event0))))
(defrule RNotificationServicePolicy3
(event (service ?service0&NotificationService|Any) (type ?type0) (name ?event0&ProcessEvent))
=>(assert (event (service Any) (type Policy) (name RNotificationServicePolicy3Event1C))))
```

Notification Service Runtime Trace

```
#First request
Sending Event InputEvent
Sending Event ProcessEvent
[Login] Reading user list from file
Sending Event InputEvent
username = Susan.Smith
password = 1234
membership = Gold
Sending Event ProcessEvent
[Login] Susan Smith has successfully login
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event InputEvent
Sending Event ProcessEvent
Sending Event OutputEvent
Sending Event IdleEvent
[AddressBook] Reading user list from file
[AddressBook] Reading contact list from file
Sending Event InputEvent
Sending Event ProcessEvent
Receiving Actions
Action: doQuickSort
[AddressBook] Requesting search for contact for user John.Murphy
[AddressBook] $$Using Quicksort algorithm to sort contact list
Sending Event SortedEvent
Receiving Actions
Action: doBinarySearch
[AddressBook] $$Searching for 02 using Binary search algorithm
[AddressBook] Contact detail found for originator: Susan.Smith@moto.co.uk, 9894455
[AddressBook] $$Searching for 01 using Binary search algorithm
```

```
[AddressBook] Contact detail found for recipient: John.Murphy@svaley.com, 8381122
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event InputEvent
Sending Event ProcessEvent
Sending Event InputEvent
Sending Event ProcessEvent
Receiving Actions
Action: doHtmlMessage
[Email] Creating email body
[Email] $$Adding HTML tags to email body
[Email] $$Transforming message to HTML format
[Email] Adding message to email body: This month's edition of the Times magazine has arrived
[Email] Generating email header
[Email] Setting From field = Susan.Smith@moto.co.uk
[Email] Setting To field = John.Murphy@svaley.com
[Email] Setting email size
Sending Event MessageCreatedEvent
Receiving Actions
Action: doHmtlHeader
[Email] Sending email to Susan.Smith@moto.co.uk
[Email] Email Acknowledgement received
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event InputEvent
Sending Event ProcessEvent
[Phone] Creating vxml dialog to notify recipient by phone
[Phone] Calling recipient on 8381122
[Phone] User has answered the call
[Phone] Informing recipient of a notification from 9894455
[Phone] Notifying recipient with the following message: This month's edition of the Times magazine
has arrived
[Phone] User has acknowledged this message
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event OutputEvent
Sending Event IdleEvent
________________________________________
#Second request
Sending Event InputEvent
Sending Event ProcessEvent
[Login] Reading user list from file
Sending Event InputEvent
username = Susan.Smith
password = 1234
membership = Silver
Sending Event ProcessEvent
[Login] Susan Smith has successfully login
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event InputEvent
Sending Event ProcessEvent
Sending Event OutputEvent
Sending Event IdleEvent
[AddressBook] Reading user list from file
[AddressBook] Reading contact list from file
Sending Event InputEvent
Sending Event ProcessEvent
```

```
Receiving Actions
Action: useEmergencyContact
[AddressBook] Requesting search for contact for user John.Murphy
[AddressBook] $$Changing criteria to search for emergency contact address
Sending Event SortedEvent
[AddressBook] Searching for 02 using Linear search algorithm
[AddressBook] Contact detail found for originator: Susan.Smith@moto.co.uk, 9894455
[AddressBook] Searching for 01 using Linear search algorithm
[AddressBook] Contact detail found for recipient: John.Murphy@svaley.com, 8381122
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event InputEvent
Sending Event ProcessEvent
Sending Event InputEvent
Sending Event ProcessEvent
[Email] Creating email body
[Email] Adding message to email body: Your package has arrived
[Email] Generating email header
[Email] Setting From field = Susan.Smith@moto.co.uk
[Email] Setting To field = John.Murphy@svaley.com
[Email] Setting email size
Sending Event MessageCreatedEvent
[Email] Sending email to Susan.Smith@moto.co.uk
[Email] Email Acknowledgement received
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event InputEvent
Sending Event ProcessEvent
Receiving Actions
Action: setAuthentication
[Phone] Creating vxml dialog to notify recipient by phone
[Phone] $$Changing the vxml dialog to request user's pin before notifying recipient
[Phone] Calling recipient on 8381122
[Phone] User has answered the call
[Phone] $$Requesting the user for a pin
[Phone] $$Validating the pin supplied by the user - accepted
[Phone] Informing recipient of a notification from 9894455
[Phone] Notifying recipient with the following message: Your package has arrived
[Phone] User has acknowledged this message
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event OutputEvent
Sending Event IdleEvent
Sending Event OutputEvent
Sending Event IdleEvent
```

Appendix H – PhotoAlbumPrint Service Example

The artefacts produced for the PhotoAlbumPrint service example used in the thesis.

PhotoAlbumPrint Service

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE uridef[
  <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns">
  <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema">
  <!ENTITY owl "http://www.w3.org/2002/07/owl">
  <!ENTITY service "http://www.daml.org/services/owl-s/1.1/Service.owl">
  <!ENTITY profile "http://www.daml.org/services/owl-s/1.1/Profile.owl">
  <!ENTITY process "http://www.daml.org/services/owl-s/1.1/Process.owl">
  <!ENTITY grounding "http://www.daml.org/services/owl-s/1.1/Grounding.owl">
  <!ENTITY gprocess "http://www.daml.org/services/owl-s/1.1/PhotoAlbumPrintProcess.owl">
  <!ENTITY DEFAULT "http://www.daml.org/services/owl-s/1.1/PhotoAlbumPrintService.owl">
]>

<rdf:RDF
  xmlns:rdf= "&rdf;#"
  xmlns:rdfs="&rdfs;#"
  xmlns:owl = "&owl;#"
  xmlns:service= "&service;#"
  xmlns:profile= "&profile;#"
  xmlns:process= "&process;#"
  xmlns:grounding= "&grounding;#"
  xmlns:gprocess= "&gprocess;#"
  xmlns    ="&DEFAULT;#"
  xml:base="&DEFAULT;"
  >

 <owl:Ontology rdf:about="">
   <owl:versionInfo>
     $Id: CongoService.owl,v 1.25 2004/10/07 05:49:28 martin Exp $
   </owl:versionInfo>
   <rdfs:comment>
   This ontology represents the OWL-S service description for the
   Congo web service example.
   </rdfs:comment>
   <owl:imports rdf:resource="&service;" />
   <owl:imports rdf:resource="&profile;" />
   <owl:imports rdf:resource="&process;" />
   <owl:imports rdf:resource="&grounding;" />
   <owl:imports rdf:resource="&gprocess;" />
 </owl:Ontology>
<service:Service rdf:ID="PhotoAlbumPrintService">
  <!-- Reference to the Process Model -->
  <service:describedBy rdf:resource="&gprocess;#PhotoAlbumPrintProcess"/>
</service:Service>
</rdf:RDF>
```

PhotoAlbumPrint Process

```
<rdf:RDF
  xmlns:rss="http://purl.org/rss/1.0/"
  xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
  xmlns="http://www.daml.org/services/owl-s/1.1/PhotoAlbumPrintProcess.owl#"
  xmlns:profileHierarchy="http://www.daml.org/services/owl-s/1.1/ProfileHierarchy.owl#"
  xmlns:objList="http://www.daml.org/services/owl-s/1.1/generic/ObjectList.owl#"
  xmlns:swrl="http://www.w3.org/2003/11/swrl#"
  xmlns:time="http://www.isi.edu/~pan/damltime/time-entry.owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:expr="http://www.daml.org/services/owl-s/1.1/generic/Expression.owl#"
  xmlns:jms="http://jena.hpl.hp.com/2003/08/jms#"
  xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:base="http://www.daml.org/services/owl-s/1.1/PhotoAlbumPrintProcess.owl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
 xml:base="http://www.daml.org/services/owl-s/1.1/PhotoAlbumPrintProcess.owl">
 <owl:Ontology rdf:about="">
  <owl:imports>
   <owl:Ontology rdf:about="Process.owl"/>
  </owl:imports>
  <owl:imports>
   <owl:Ontology rdf:about="http://www.isi.edu/~pan/damltime/time-entry.owl"/>
  </owl:imports>
  <owl:imports>
   <owl:Ontology rdf:about="Service.owl"/>
  </owl:imports>
  <owl:versionInfo>
  $Id: CongoProcess.owl,v 1.79 2005/02/03 22:43:43 martin Exp $
 </owl:versionInfo>
  <rdfs:comment>

 </rdfs:comment>
  <owl:imports>
   <owl:Ontology rdf:about="ProfileHierarchy.owl"/>
  </owl:imports>
 </owl:Ontology>
 <process:Input rdf:ID="AlbumPhotos">
  <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
  >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
 </process:Input>
 <process:Input rdf:ID="PhotoSize">
  <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
  >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
 </process:Input>
 <process:AtomicProcess rdf:ID="PhotoAlbumService">
  <process:hasInput rdf:resource="#AlbumPhotos"/>
  <process:hasOutput>
   <process:Output rdf:ID="PhotoAlbum">
    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
   </process:Output>
  </process:hasOutput>
  <process:hasOutput>
   <process:Output rdf:ID="PhotoAlbumSize">
```

```
        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/2001/XMLSchema#decimal</process:parameterType>
      </process:Output>
    </process:hasOutput>
    <process:hasOutput>
      <process:Output rdf:ID="PhotoAlbumColours">
        <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
        >http://www.w3.org/2001/XMLSchema#decimal</process:parameterType>
      </process:Output>
    </process:hasOutput>
    <process:hasFiniteStateMachine
rdf:resource="PhotoAlbumServiceFSM.owl#PhotoAlbumServiceFSM"/>
  </process:AtomicProcess>
  <process:Input rdf:ID="PhotoCategory">
    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
  </process:Input>
  <process:Input rdf:ID="NumberOfPages">
    <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
    >http://www.w3.org/2001/XMLSchema#decimal</process:parameterType>
  </process:Input>
  <process:Perform rdf:ID="PerformPhotoService">
    <process:process>
      <process:AtomicProcess rdf:ID="PhotoService">
        <process:hasInput>
          <process:Input rdf:ID="RawPhotos">
            <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
            >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
          </process:Input>
        </process:hasInput>
        <process:hasInput rdf:resource="#PhotoCategory"/>
        <process:hasInput rdf:resource="#PhotoSize"/>
        <process:hasOutput>
          <process:Output rdf:ID="ProcessedPhotos">
            <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
            >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
          </process:Output>
        </process:hasOutput>
        <process:hasFiniteStateMachine rdf:resource="PhotoServiceFSM.owl#PhotoServiceFSM"/>
      </process:AtomicProcess>
    </process:process>
    <process:hasDataFrom>
      <process:InputBinding>
        <process:toParam rdf:resource="#RawPhotos"/>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#CameraPhotos"/>
            <process:fromProcess rdf:resource="http://www.daml.org/services/owl-
s/1.1/Process.owl#TheParentPerform"/>
          </process:ValueOf>
        </process:valueSource>
      </process:InputBinding>
    </process:hasDataFrom>
    <process:hasDataFrom>
      <process:InputBinding>
        <process:toParam rdf:resource="#PhotoCategory"/>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#CameraPhotoCategory"/>
          <process:fromProcess rdf:resource="http://www.daml.org/services/owl-
s/1.1/Process.owl#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:toParam rdf:resource="#PhotoSize"/>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#CameraPhotoSize"/>
          <process:fromProcess rdf:resource="http://www.daml.org/services/owl-
s/1.1/Process.owl#TheParentPerform"/>
        </process:ValueOf>
      </process:valueSource>
    </process:InputBinding>
  </process:hasDataFrom>
</process:Perform>
<process:Input rdf:ID="CameraPhotoCategory">
  <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
  >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
</process:Input>
<process:Perform rdf:ID="PerformPrintService">
  <process:process>
    <process:AtomicProcess rdf:ID="PrintService">
      <process:hasInput>
        <process:Input rdf:ID="DocumentName">
          <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
        </process:Input>
      </process:hasInput>
      <process:hasInput rdf:resource="#NumberOfPages"/>
      <process:hasInput>
        <process:Input rdf:ID="NumberOfColours">
          <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.w3.org/2001/XMLSchema#decimal</process:parameterType>
        </process:Input>
      </process:hasInput>
      <process:hasOutput>
        <process:Output rdf:ID="NumberOfSheets">
          <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
          >http://www.w3.org/2001/XMLSchema#boolean</process:parameterType>
        </process:Output>
      </process:hasOutput>
      <process:hasFiniteStateMachine rdf:resource="PrintServiceFSM.owl#PrintServiceFSM"/>
    </process:AtomicProcess>
  </process:process>
  <process:hasDataFrom>
    <process:InputBinding>
      <process:toParam rdf:resource="#DocumentName"/>
      <process:valueSource>
        <process:ValueOf>
          <process:theVar rdf:resource="#PhotoAlbum"/>
          <process:fromProcess rdf:resource="#PerformPhotoAlbumService"/>
        </process:ValueOf>
      </process:valueSource>
    </process:InputBinding>
  </process:hasDataFrom>
  <process:hasDataFrom>
```

```
      <process:InputBinding>
        <process:toParam rdf:resource="#NumberOfPages"/>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#PhotoAlbumSize"/>
            <process:fromProcess rdf:resource="#PerformPhotoAlbumService"/>
          </process:ValueOf>
        </process:valueSource>
      </process:InputBinding>
    </process:hasDataFrom>
    <process:hasDataFrom>
      <process:InputBinding>
        <process:toParam rdf:resource="#NumberOfColours"/>
        <process:valueSource>
          <process:ValueOf>
            <process:theVar rdf:resource="#PhotoAlbumColours"/>
            <process:fromProcess rdf:resource="#PerformPhotoAlbumService"/>
          </process:ValueOf>
        </process:valueSource>
      </process:InputBinding>
    </process:hasDataFrom>
  </process:Perform>
  <process:CompositeProcess rdf:ID="PhotoAlbumPrintProcess">
   <process:hasInput>
    <process:Input rdf:ID="CameraPhotos">
     <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
     >http://www.w3.org/2001/XMLSchema#string</process:parameterType>
    </process:Input>
   </process:hasInput>
   <process:hasInput>
    <process:Input rdf:ID="CameraPhotoSize">
     <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
     >http://www.w3.org/2001/XMLSchema#decimal</process:parameterType>
    </process:Input>
   </process:hasInput>
   <process:hasInput rdf:resource="#CameraPhotoCategory"/>
   <process:hasOutput>
    <process:Output rdf:ID="AlbumSize">
     <process:parameterType rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"
     >http://www.w3.org/2001/XMLSchema#decimal</process:parameterType>
    </process:Output>
   </process:hasOutput>
   <process:composedOf>
    <process:Sequence>
     <process:components>
      <process:ControlConstructList>
       <objList:first rdf:resource="#PerformPhotoService"/>
       <objList:rest>
        <process:ControlConstructList>
         <objList:first>
          <process:Perform rdf:ID="PerformPhotoAlbumService">
           <process:process rdf:resource="#PhotoAlbumService"/>
            <process:hasDataFrom>
             <process:InputBinding>
              <process:toParam rdf:resource="#AlbumPhotos"/>
              <process:valueSource>
               <process:ValueOf>
                <process:heVar rdf:resource="#ProcessedPhotos"/>
                <process:fromProcess rdf:resource="#PerformPhotoService"/>
               </process:ValueOf>
```

```
                </process:valueSource>
              </process:InputBinding>
            </process:hasDataFrom>
            </process:Perform>
          </objList:first>
          <objList:rest>
            <process:ControlConstructList>
              <objList:first rdf:resource="#PerformPrintService"/>
              <objList:rest rdf:resource="generic/ObjectList.owl#nil"/>
            </process:ControlConstructList>
          </objList:rest>
        </process:ControlConstructList>
      </objList:rest>
    </process:ControlConstructList>
   </process:components>
  </process:Sequence>
 </process:composedOf>
 <process:hasFiniteStateMachine
rdf:resource="PhotoAlbumPrintProcessFSM.owl#PhotoAlbumPrintProcessFSM"/>
 </process:CompositeProcess>
</rdf:RDF>
```

PhotoAlbumPrint Service's FSM

```
<rdf:RDF
  xmlns:fsm="http://kdeg.cs.tcd.ie/FiniteStateMachine.owl#"
  xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
  xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
  xmlns:policy="http://kdeg.cs.tcd.ie/Policy.owl#"
  xmlns:grounding="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
  xmlns="http://www.daml.org/services/owl-s/1.1/PhotoAlbumPrintProcessFSM.owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:base="http://www.daml.org/services/owl-s/1.1/PhotoAlbumPrintProcessFSM.owl"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
  xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
 xml:base="http://www.daml.org/services/owl-s/1.1/PhotoAlbumPrintProcessFSM.owl">
 <owl:Ontology rdf:about="">
  <owl:imports rdf:resource="http://kdeg.cs.tcd.ie/FiniteStateMachine.owl"/>
 </owl:Ontology>
 <fsm:Transition rdf:ID="BlackWhitePhotoInitialStateInitialTransition">
  <fsm:trigger>
   <fsm:SignalEvent rdf:ID="InitialEvent">
    <fsm:signal>
     <fsm:Signal rdf:ID="InitialEventSource">
      <fsm:value>InitialSignal</fsm:value>
     </fsm:Signal>
    </fsm:signal>
   </fsm:SignalEvent>
  </fsm:trigger>
  <fsm:source>
   <fsm:PseudoState rdf:ID="BlackWhitePhotoInitialState">
    <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
    <fsm:outgoing rdf:resource="#BlackWhitePhotoInitialStateInitialTransition"/>
   </fsm:PseudoState>
  </fsm:source>
  <fsm:target>
   <fsm:FinalState rdf:ID="BlackWhitePhotoState1">
   <fsm:doActivity>setColourMode(false)</fsm:doActivity>
   <fsm:incoming rdf:resource="#BlackWhitePhotoInitialStateInitialTransition"/>
  </fsm:FinalState>
 </fsm:target>
</fsm:Transition>
<fsm:Transition rdf:ID="IdleToInput">
 <fsm:trigger>
  <fsm:SignalEvent rdf:ID="InputEvent">
   <fsm:signal>
    <fsm:Signal rdf:ID="InputEventSource">
     <fsm:value>InputSignal</fsm:value>
    </fsm:Signal>
   </fsm:signal>
  </fsm:SignalEvent>
 </fsm:trigger>
 <fsm:source>
  <fsm:CompositeState rdf:ID="IdleState">
   <fsm:incoming>
    <fsm:Transition rdf:ID="InitialT">
     <fsm:trigger rdf:resource="#InitialEvent"/>
     <fsm:source>
      <fsm:PseudoState rdf:ID="InitialState">
       <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
       <fsm:outgoing rdf:resource="#InitialT"/>
      </fsm:PseudoState>
     </fsm:source>
     <fsm:target rdf:resource="#IdleState"/>
    </fsm:Transition>
   </fsm:incoming>
   <fsm:outgoing rdf:resource="#IdleToInput"/>
   <fsm:incoming>
    <fsm:Transition rdf:ID="OutputToIdle">
     <fsm:trigger>
      <fsm:SignalEvent rdf:ID="IdleEvent">
       <fsm:signal>
        <fsm:Signal rdf:ID="IdleEventSource">
         <fsm:value>IdleSignal</fsm:value>
        </fsm:Signal>
       </fsm:signal>
      </fsm:SignalEvent>
     </fsm:trigger>
     <fsm:source>
      <fsm:CompositeState rdf:ID="OutputState">
       <fsm:subvertex>
        <fsm:SubmachineState rdf:ID="ColourPrintingSubSM">
         <fsm:submachine>
          <fsm:StateMachine rdf:ID="ColourPrinting">
           <fsm:top>
            <fsm:CompositeState rdf:ID="ColourPrintingCS">
             <fsm:subvertex>
              <fsm:PseudoState rdf:ID="ColourPrintingInitialState">
               <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
               <fsm:outgoing>
                <fsm:Transition rdf:ID="ColourPrintingInitialStateInitialTransition">
                 <fsm:trigger rdf:resource="#InitialEvent"/>
                 <fsm:source rdf:resource="#ColourPrintingInitialState"/>
                 <fsm:target>
                  <fsm:SimpleState rdf:ID="ColourPrintingState1">
                   <fsm:doActivity>checkcartage(type_colour)</fsm:doActivity>
                   <fsm:incoming rdf:resource="#ColourPrintingInitialStateInitialTransition"/>
```

```
                    <fsm:outgoing>
                     <fsm:Transition rdf:ID="ColourPrintingState1Transition1">
                      <fsm:trigger>
                       <fsm:SignalEvent rdf:ID="Printing">
                        <fsm:signal>
                         <fsm:Signal rdf:ID="PrintingSource">
                          <fsm:value
                          >PrintingSignal</fsm:value>
                         </fsm:Signal>
                        </fsm:signal>
                       </fsm:SignalEvent>
                      </fsm:trigger>
                      <fsm:source rdf:resource="#ColourPrintingState1"/>
                      <fsm:target>
                       <fsm:FinalState rdf:ID="ColourPrintingState2">
                        <fsm:doActivity
                        >usecartage(type_colour)</fsm:doActivity>
                        <fsm:incoming rdf:resource="#ColourPrintingState1Transition1"/>
                       </fsm:FinalState>
                      </fsm:target>
                      <fsm:guard>
                       <fsm:Guard rdf:ID="ColourPrintingState1Transition1Guard">
                        <fsm:expression>
                         <fsm:Expression
rdf:ID="ColourPrintingState1Transition1Expression">
                          <fsm:subject>
                           <fsm:Subject rdf:ID="colourcartage"/>
                          </fsm:subject>
                          <fsm:predicate>
                           <fsm:Predicate rdf:ID="equal"/>
                          </fsm:predicate>
                          <fsm:value
                          >present</fsm:value>
                         </fsm:Expression>
                        </fsm:expression>
                       </fsm:Guard>
                      </fsm:guard>
                     </fsm:Transition>
                    </fsm:outgoing>
                   </fsm:SimpleState>
                  </fsm:target>
                 </fsm:Transition>
                </fsm:outgoing>
               </fsm:PseudoState>
              </fsm:subvertex>
              <fsm:subvertex rdf:resource="#ColourPrintingState1"/>
              <fsm:subvertex rdf:resource="#ColourPrintingState2"/>
             </fsm:CompositeState>
            </fsm:top>
            <fsm:comment>Set the printer to print in colour</fsm:comment>
           </fsm:StateMachine>
          </fsm:submachine>
         </fsm:SubmachineState>
        </fsm:subvertex>
        <fsm:incoming>
         <fsm:Transition rdf:ID="ProcessToOutput">
          <fsm:trigger>
           <fsm:SignalEvent rdf:ID="OutputEvent">
            <fsm:signal>
             <fsm:Signal rdf:ID="OutputEventSource">
```

```
                <fsm:value>OutputSignal</fsm:value>
              </fsm:Signal>
            </fsm:signal>
          </fsm:SignalEvent>
        </fsm:trigger>
        <fsm:source>
         <fsm:CompositeState rdf:ID="ProcessState">
          <fsm:subvertex>
           <fsm:SubmachineState rdf:ID="CreatePostcardSubSM">
            <fsm:submachine>
             <fsm:StateMachine rdf:ID="CreatePostcard">
              <fsm:top>
               <fsm:CompositeState rdf:ID="CreatePostcardCS">
                <fsm:subvertex>
                 <fsm:PseudoState rdf:ID="CreatePostcardInitialState">
                  <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
                 </fsm:PseudoState>
                </fsm:subvertex>
                <fsm:subvertex>
                 <fsm:SimpleState rdf:ID="CreatePostcardState1">
                  <fsm:doActivity>createPostcard()</fsm:doActivity>
                 </fsm:SimpleState>
                </fsm:subvertex>
               </fsm:CompositeState>
              </fsm:top>
              <fsm:comment>Creates an album of postcards with the given
photos</fsm:comment>
             </fsm:StateMachine>
            </fsm:submachine>
           </fsm:SubmachineState>
          </fsm:subvertex>
          <fsm:subvertex>
           <fsm:SubmachineState rdf:ID="ExpensiveModeSubSM">
            <fsm:submachine>
             <fsm:StateMachine rdf:ID="ExpensiveMode">
              <fsm:top>
               <fsm:CompositeState rdf:ID="ExpensiveModeCS">
                <fsm:subvertex>
                 <fsm:PseudoState rdf:ID="ExpensiveModeInitialState">
                  <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
                  <fsm:outgoing>
                   <fsm:Transition rdf:ID="ExpensiveModeInitialStateInitialTransition">
                    <fsm:trigger rdf:resource="#InitialEvent"/>
                    <fsm:source rdf:resource="#ExpensiveModeInitialState"/>
                    <fsm:target>
                     <fsm:FinalState rdf:ID="ExpensiveModeState1">
                      <fsm:doActivity
                      >printmode(1, singlesided)</fsm:doActivity>
                      <fsm:incoming
rdf:resource="#ExpensiveModeInitialStateInitialTransition"/>
                     </fsm:FinalState>
                    </fsm:target>
                   </fsm:Transition>
                  </fsm:outgoing>
                 </fsm:PseudoState>
                </fsm:subvertex>
                <fsm:subvertex rdf:resource="#ExpensiveModeState1"/>
               </fsm:CompositeState>
              </fsm:top>
              <fsm:comment>prints a page per sheet</fsm:comment>
```

```
               </fsm:StateMachine>
              </fsm:submachine>
             </fsm:SubmachineState>
            </fsm:subvertex>
            <fsm:outgoing rdf:resource="#ProcessToOutput"/>
            <fsm:subvertex>
             <fsm:SubmachineState rdf:ID="HighPhotoQualitySubSM">
              <fsm:submachine>
               <fsm:StateMachine rdf:ID="HighPhotoQuality">
                <fsm:top>
                 <fsm:CompositeState rdf:ID="HighPhotoQualityCS">
                  <fsm:subvertex>
                   <fsm:PseudoState rdf:ID="HighPhotoQualityInitialState">
                    <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
                    <fsm:outgoing>
                     <fsm:Transition rdf:ID="HighPhotoQualityInitialStateInitialTransition">
                      <fsm:trigger rdf:resource="#InitialEvent"/>
                      <fsm:source rdf:resource="#HighPhotoQualityInitialState"/>
                      <fsm:target>
                       <fsm:FinalState rdf:ID="HighPhotoQualityState1">
                        <fsm:doActivity
                        >setResolution(high)</fsm:doActivity>
                        <fsm:incoming
rdf:resource="#HighPhotoQualityInitialStateInitialTransition"/>
                       </fsm:FinalState>
                      </fsm:target>
                     </fsm:Transition>
                    </fsm:outgoing>
                   </fsm:PseudoState>
                  </fsm:subvertex>
                  <fsm:subvertex rdf:resource="#HighPhotoQualityState1"/>
                 </fsm:CompositeState>
                </fsm:top>
                <fsm:comment>Produce high quality photos</fsm:comment>
               </fsm:StateMachine>
              </fsm:submachine>
             </fsm:SubmachineState>
            </fsm:subvertex>
            <fsm:subvertex>
             <fsm:SubmachineState rdf:ID="IntermediateModeSubSM">
              <fsm:submachine>
               <fsm:StateMachine rdf:ID="IntermediateMode">
                <fsm:top>
                 <fsm:CompositeState rdf:ID="IntermediateModeCS">
                  <fsm:subvertex>
                   <fsm:PseudoState rdf:ID="IntermediateModeInitialState">
                    <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
                    <fsm:outgoing>
                     <fsm:Transition rdf:ID="IntermediateModeInitialStateInitialTransition">
                      <fsm:trigger rdf:resource="#InitialEvent"/>
                      <fsm:source rdf:resource="#IntermediateModeInitialState"/>
                      <fsm:target>
                       <fsm:FinalState rdf:ID="IntermediateModeState1">
                        <fsm:doActivity
                        >printmode(1, doublesided)</fsm:doActivity>
                        <fsm:incoming
rdf:resource="#IntermediateModeInitialStateInitialTransition"/>
                       </fsm:FinalState>
                      </fsm:target>
                     </fsm:Transition>
```

```
                          </fsm:outgoing>
                        </fsm:PseudoState>
                      </fsm:subvertex>
                      <fsm:subvertex rdf:resource="#IntermediateModeState1"/>
                    </fsm:CompositeState>
                  </fsm:top>
                  <fsm:comment>prints 2 pages per sheet</fsm:comment>
                </fsm:StateMachine>
              </fsm:submachine>
            </fsm:SubmachineState>
          </fsm:subvertex>
          <fsm:subvertex>
            <fsm:SubmachineState rdf:ID="CreateCalendarSubSM">
              <fsm:submachine>
                <fsm:StateMachine rdf:ID="CreateCalendar">
                  <fsm:top>
                    <fsm:CompositeState rdf:ID="CreateCalendarCS">
                      <fsm:subvertex>
                        <fsm:PseudoState rdf:ID="CreateCalendarInitialState">
                          <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
                          <fsm:outgoing>
                            <fsm:Transition rdf:ID="CreateCalendarInitialStateInitialTransition">
                              <fsm:trigger rdf:resource="#InitialEvent"/>
                              <fsm:source rdf:resource="#CreateCalendarInitialState"/>
                              <fsm:target>
                                <fsm:SimpleState rdf:ID="CreateCalendarState1">
                                  <fsm:doActivity
                                  >createCalendar()</fsm:doActivity>
                                  <fsm:incoming
rdf:resource="#CreateCalendarInitialStateInitialTransition"/>
                                </fsm:SimpleState>
                              </fsm:target>
                            </fsm:Transition>
                          </fsm:outgoing>
                        </fsm:PseudoState>
                      </fsm:subvertex>
                      <fsm:subvertex rdf:resource="#CreateCalendarState1"/>
                    </fsm:CompositeState>
                  </fsm:top>
                  <fsm:comment>creates a calendar with the first 12 photos</fsm:comment>
                </fsm:StateMachine>
              </fsm:submachine>
            </fsm:SubmachineState>
          </fsm:subvertex>
          <fsm:subvertex>
            <fsm:SubmachineState rdf:ID="EconomyModeSubSM">
              <fsm:submachine>
                <fsm:StateMachine rdf:ID="EconomyMode">
                  <fsm:top>
                    <fsm:CompositeState rdf:ID="EconomyModeCS">
                      <fsm:subvertex>
                        <fsm:PseudoState rdf:ID="EconomyModeInitialState">
                          <fsm:pseudoStateKind>initial</fsm:pseudoStateKind>
                          <fsm:outgoing>
                            <fsm:Transition rdf:ID="EconomyModeInitialStateInitialTransition">
                              <fsm:trigger rdf:resource="#InitialEvent"/>
                              <fsm:source rdf:resource="#EconomyModeInitialState"/>
                              <fsm:target>
                                <fsm:FinalState rdf:ID="EconomyModeState1">
                                  <fsm:doActivity
                        >printmode(2, doublesided)</fsm:doActivity>
                        <fsm:incoming
rdf:resource="#EconomyModeInitialStateInitialTransition"/>
                       </fsm:FinalState>
                      </fsm:target>
                     </fsm:Transition>
                    </fsm:outgoing>
                   </fsm:PseudoState>
                  </fsm:subvertex>
                  <fsm:subvertex rdf:resource="#EconomyModeState1"/>
                 </fsm:CompositeState>
                </fsm:top>
                <fsm:comment>prints 4 pages per sheet</fsm:comment>
               </fsm:StateMachine>
              </fsm:submachine>
             </fsm:SubmachineState>
            </fsm:subvertex>
            <fsm:subvertex>
             <fsm:SubmachineState rdf:ID="BlackWhitePhotoSubSM">
              <fsm:submachine>
               <fsm:StateMachine rdf:ID="BlackWhitePhoto">
                <fsm:top>
                 <fsm:CompositeState rdf:ID="BlackWhitePhotoCS">
                  <fsm:subvertex rdf:resource="#BlackWhitePhotoInitialState"/>
                  <fsm:subvertex rdf:resource="#BlackWhitePhotoState1"/>
                 </fsm:CompositeState>
                </fsm:top>
                <fsm:comment>Produce black and white photos</fsm:comment>
               </fsm:StateMachine>
              </fsm:submachine>
             </fsm:SubmachineState>
            </fsm:subvertex>
            <fsm:incoming>
             <fsm:Transition rdf:ID="InputToProcess">
              <fsm:trigger>
               <fsm:SignalEvent rdf:ID="ProcessEvent">
                <fsm:signal>
                 <fsm:Signal rdf:ID="ProcessEventSource">
                  <fsm:value>ProcessSignal</fsm:value>
                 </fsm:Signal>
                </fsm:signal>
               </fsm:SignalEvent>
              </fsm:trigger>
              <fsm:source>
               <fsm:CompositeState rdf:ID="InputState">
                <fsm:subvertex>
                 <fsm:SubmachineState rdf:ID="RemoveRedEyeSubSM">
                  <fsm:submachine>
                   <fsm:StateMachine rdf:ID="RemoveRedEye">
                    <fsm:top>
                     <fsm:CompositeState rdf:ID="RemoveRedEyeCS">
                      <fsm:subvertex>
                       <fsm:PseudoState rdf:ID="RemoveRedEyeInitialState">
                        <fsm:pseudoStateKind
                        >initial</fsm:pseudoStateKind>
                        <fsm:outgoing>
                         <fsm:Transition rdf:ID="RemoveRedEyeInitialStateInitialTransition">
                          <fsm:trigger rdf:resource="#InitialEvent"/>
                          <fsm:source rdf:resource="#RemoveRedEyeInitialState"/>
                          <fsm:target>
            <fsm:SimpleState rdf:ID="RemoveRedEyeState1">
             <fsm:doActivity
             >searchRedEye()</fsm:doActivity>
             <fsm:incoming
rdf:resource="#RemoveRedEyeInitialStateInitialTransition"/>
             <fsm:outgoing>
              <fsm:Transition rdf:ID="RemoveRedEyeState1Transition1">
               <fsm:trigger>
                <fsm:SignalEvent rdf:ID="ProcessingPhoto">
                 <fsm:signal>
                  <fsm:Signal rdf:ID="ProcessingPhotoSource">
                   <fsm:value>ProcessingSignal</fsm:value>
                  </fsm:Signal>
                 </fsm:signal>
                </fsm:SignalEvent>
               </fsm:trigger>
               <fsm:source rdf:resource="#RemoveRedEyeState1"/>
               <fsm:target>
                <fsm:FinalState rdf:ID="RemoveRedEyeState2">
                 <fsm:doActivity>removeRedEye()</fsm:doActivity>
                 <fsm:incoming
rdf:resource="#RemoveRedEyeState1Transition1"/>
                </fsm:FinalState>
               </fsm:target>
               <fsm:guard>
                <fsm:Guard rdf:ID="RemoveRedEyeState1Transition1Guard">
                 <fsm:expression>
             <fsm:Expression
rdf:ID="RemoveRedEyeState1Transition1Expression">
              <fsm:subject>
               <fsm:Subject rdf:ID="redeyelocation"/>
              </fsm:subject>
              <fsm:predicate>
               <fsm:Predicate rdf:ID="inequal"/>
              </fsm:predicate>
              <fsm:value>0</fsm:value>
              </fsm:Expression>
             </fsm:expression>
               </fsm:Guard>
               </fsm:guard>
              </fsm:Transition>
             </fsm:outgoing>
            </fsm:SimpleState>
           </fsm:target>
          </fsm:Transition>
         </fsm:outgoing>
        </fsm:PseudoState>
       </fsm:subvertex>
       <fsm:subvertex rdf:resource="#RemoveRedEyeState1"/>
       <fsm:subvertex rdf:resource="#RemoveRedEyeState2"/>
      </fsm:CompositeState>
     </fsm:top>
     <fsm:comment>Removes red eyes in the photos</fsm:comment>
    </fsm:StateMachine>
   </fsm:submachine>
  </fsm:SubmachineState>
 </fsm:subvertex>
 <fsm:incoming rdf:resource="#IdleToInput"/>
 <fsm:outgoing rdf:resource="#InputToProcess"/>
</fsm:CompositeState>
```

```
                  </fsm:source>
                  <fsm:target rdf:resource="#ProcessState"/>
                 </fsm:Transition>
                </fsm:incoming>
               </fsm:CompositeState>
              </fsm:source>
              <fsm:target rdf:resource="#OutputState"/>
             </fsm:Transition>
            </fsm:incoming>
            <fsm:outgoing rdf:resource="#OutputToIdle"/>
           </fsm:CompositeState>
          </fsm:source>
          <fsm:target rdf:resource="#IdleState"/>
         </fsm:Transition>
        </fsm:incoming>
      </fsm:CompositeState>
     </fsm:source>
     <fsm:target rdf:resource="#InputState"/>
   </fsm:Transition>
   <fsm:StateMachine rdf:ID="PhotoAlbumPrintProcessFSM">
     <fsm:top>
       <fsm:CompositeState rdf:ID="ServiceState">
         <fsm:subvertex rdf:resource="#InitialState"/>
         <fsm:subvertex rdf:resource="#IdleState"/>
         <fsm:subvertex rdf:resource="#ProcessState"/>
         <fsm:subvertex rdf:resource="#InputState"/>
         <fsm:subvertex rdf:resource="#OutputState"/>
       </fsm:CompositeState>
     </fsm:top>
   </fsm:StateMachine>
</rdf:RDF>
```

PhotoAlbumPrint Service's Management Policy

```
<rdf:RDF
   xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
   xmlns:policy="http://kdeg.cs.tcd.ie/Policy.owl#"
   xmlns:fsm="http://kdeg.cs.tcd.ie/FiniteStateMachine.owl#"
   xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
   xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
   xmlns:base="http://www.daml.org/services/owl-s/1.1/PhotoAlbumPrintServicePolicy.owl"
   xmlns:grounding="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
   xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:thisprocess="http://www.daml.org/services/owl-s/1.1/PhotoAlbumPrintProcess.owl#"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns="http://www.daml.org/services/owl-s/1.1/PhotoAlbumPrintServicePolicy.owl#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xml:base="http://www.daml.org/services/owl-s/1.1/PhotoAlbumPrintServicePolicy.owl">
  <owl:Ontology rdf:about="">
    <owl:imports rdf:resource="http://kdeg.cs.tcd.ie/Policy.owl"/>
    <owl:imports rdf:resource="PhotoAlbumPrint.owl"/>
  </owl:Ontology>
  <policy:ComplexAction rdf:ID="PhotoAlbumPrintPolicy1Action1">
    <rdfs:first>
      <policy:AndList>
        <rdfs:first>
          <policy:SimpleAction rdf:ID="PhotoAlbumPrintPolicy1Action101">
            <policy:value>RemoveRedEye</policy:value>
        </policy:SimpleAction>
      </rdfs:first>
      <rdfs:rest>
        <policy:SimpleAction rdf:ID="PhotoAlbumPrintPolicy1Action102">
          <policy:value>CreateCalendar</policy:value>
        </policy:SimpleAction>
      </rdfs:rest>
    </policy:AndList>
  </rdfs:first>
  <rdfs:rest></rdfs:rest>
</policy:ComplexAction>
<policy:Subject rdf:ID="CameraPhotoCategory"/>
<policy:SimpleCondition rdf:ID="PhotoAlbumPrintPolicy1Condition1">
  <policy:subject rdf:resource="#CameraPhotoCategory"/>
  <policy:predicate>
    <policy:Predicate rdf:ID="equal"/>
  </policy:predicate>
  <policy:value>portrait</policy:value>
</policy:SimpleCondition>
<policy:Policy rdf:ID="PhotoAlbumPrintPolicy1">
  <rdfs:comment>Policy for this service to use removeredeye and create calendar adaptive behaviours
for processing portrait photos</rdfs:comment>
  <policy:target rdf:resource="PhotoAlbumPrint.owl#PhotoAlbumPrint"/>
  <policy:event>
    <policy:SimpleEvent rdf:ID="PhotoAlbumPrintPolicy1Event1">
      <policy:value>ProcessEvent</policy:value>
    </policy:SimpleEvent>
  </policy:event>
  <policy:condition rdf:resource="#PhotoAlbumPrintPolicy1Condition1"/>
  <policy:action rdf:resource="#PhotoAlbumPrintPolicy1Action1"/>
</policy:Policy>
</rdf:RDF>
```

PhotoAlbumPrint Service's Refined Policies

```
<rdf:RDF
  xmlns:process="http://www.daml.org/services/owl-s/1.1/Process.owl#"
  xmlns:policy="http://kdeg.cs.tcd.ie/Policy.owl#"
  xmlns:fsm="http://kdeg.cs.tcd.ie/FiniteStateMachine.owl#"
  xmlns:service="http://www.daml.org/services/owl-s/1.1/Service.owl#"
  xmlns:profile="http://www.daml.org/services/owl-s/1.1/Profile.owl#"
  xmlns:grounding="http://www.daml.org/services/owl-s/1.1/Grounding.owl#"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:base="http://www.daml.org/services/owl-s/1.1/PhotoAlbumPrintServiceRefinedPolicy.owl"
  xmlns:thisprocess="http://www.daml.org/services/owl-s/1.1/PhotoAlbumPrintProcess.owl#"
  xmlns="http://www.daml.org/services/owl-s/1.1/PhotoAlbumPrintServiceRefinedPolicy.owl#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xml:base="http://www.daml.org/services/owl-s/1.1/PhotoAlbumPrintServiceRefinedPolicy.owl">
 <owl:Ontology rdf:about="">
  <owl:imports rdf:resource="http://kdeg.cs.tcd.ie/Policy.owl"/>
  <owl:imports rdf:resource="PhotoAlbumPrintService.owl"/>
 </owl:Ontology>
 <policy:Subject rdf:ID="PhotoCategory"/>
 <policy:SimpleAction rdf:ID="PhotoRemoveRedEyeAction01">
  <policy:value>searchRedEye()</policy:value>
 </policy:SimpleAction>
 <policy:SimpleAction rdf:ID="PhotoRemoveRedEyeAction02">
```

```
  <policy:value>event(RemoveRedEyeState1Event)</policy:value>
</policy:SimpleAction>
<policy:SimpleCondition rdf:ID="PhotoRemoveRedEyeCondition2">
  <policy:subject>
   <policy:Subject rdf:ID="redeyelocation"/>
  </policy:subject>
  <policy:predicate>
   <policy:Predicate rdf:ID="inequal"/>
  </policy:predicate>
  <policy:value>0</policy:value>
</policy:SimpleCondition>
<policy:SimpleEvent rdf:ID="PhotoAlbumCreateCalendarEvent1b">
  <policy:value>ProcessEvent</policy:value>
</policy:SimpleEvent>
<policy:Policy rdf:ID="PhotoAlbumCreateCalendarPolicy1">
  <rdfs:comment></rdfs:comment>
  <policy:target rdf:resource="PhotoAlbumPrint.owl#PhotoAlbum"/>
  <policy:event>
   <policy:ComplexEvent rdf:ID="PhotoAlbumCreateCalendarEvent1a">
     <rdfs:first>
      <policy:AndList>
        <rdfs:first rdf:resource="#PhotoAlbumCreateCalendarEvent1b"/>
        <rdfs:rest>
         <policy:SimpleAction rdf:ID="PhotoAlbumCreateCalendarEvent1">
           <policy:value>PhotoAlbumCreateCalendarEvent</policy:value>
         </policy:SimpleAction>
        </rdfs:rest>
      </policy:AndList>
     </rdfs:first>
     <rdfs:rest></rdfs:rest>
   </policy:ComplexEvent>
  </policy:event>
  <policy:action>
   <policy:SimpleAction rdf:ID="PhotoAlbumCreateCalendarAction1">
    <policy:value>createCalendar()</policy:value>
   </policy:SimpleAction>
  </policy:action>
</policy:Policy>
<policy:ComplexAction rdf:ID="PhotoRemoveRedEyeAction1">
  <rdfs:first>
   <policy:AndList>
     <rdfs:first rdf:resource="#PhotoRemoveRedEyeAction01"/>
     <rdfs:rest rdf:resource="#PhotoRemoveRedEyeAction02"/>
   </policy:AndList>
  </rdfs:first>
  <rdfs:rest></rdfs:rest>
</policy:ComplexAction>
<policy:SimpleAction rdf:ID="PhotoRemoveRedEyeAction2">
  <policy:value>removeRedEye()</policy:value>
</policy:SimpleAction>
<policy:SimpleAction rdf:ID="PhotoRemoveRedEyeEvent1">
  <policy:value>PhotoRemoveRedEyeEvent</policy:value>
</policy:SimpleAction>
<policy:ComplexEvent rdf:ID="PhotoRemoveRedEyeEvent2">
  <rdfs:first>
   <policy:AndList>
     <rdfs:first>
      <policy:SimpleEvent rdf:ID="PhotoRemoveRedEyeEvent001">
        <policy:value>ProcessingPhoto</policy:value>
      </policy:SimpleEvent>
```

```
        </rdfs:first>
        <rdfs:rest>
          <policy:SimpleAction rdf:ID="PhotoRemoveRedEyeEvent002">
            <policy:value>RemoveRedEyeState1Event</policy:value>
          </policy:SimpleAction>
        </rdfs:rest>
      </policy:AndList>
    </rdfs:first>
    <rdfs:rest></rdfs:rest>
  </policy:ComplexEvent>
  <policy:Predicate rdf:ID="equal"/>
  <policy:Policy rdf:ID="PhotoRemoveRedEyePolicy2">
    <rdfs:comment></rdfs:comment>
    <policy:target rdf:resource="PhotoAlbumPrint.owl#Photo"/>
    <policy:event rdf:resource="#PhotoRemoveRedEyeEvent2"/>
    <policy:condition rdf:resource="#PhotoRemoveRedEyeCondition2"/>
    <policy:action rdf:resource="#PhotoRemoveRedEyeAction2"/>
  </policy:Policy>
  <policy:Policy rdf:ID="PhotoRemoveRedEyePolicy1">
    <rdfs:comment></rdfs:comment>
    <policy:target rdf:resource="PhotoAlbumPrint.owl#Photo"/>
    <policy:event>
      <policy:ComplexEvent rdf:ID="PhotoRemoveRedEyeEvent1a">
        <rdfs:first>
          <policy:AndList>
            <rdfs:first>
              <policy:SimpleEvent rdf:ID="PhotoRemoveRedEyeEvent1b">
                <policy:value>InputEvent</policy:value>
              </policy:SimpleEvent>
            </rdfs:first>
            <rdfs:rest rdf:resource="#PhotoRemoveRedEyeEvent1"/>
          </policy:AndList>
        </rdfs:first>
        <rdfs:rest></rdfs:rest>
      </policy:ComplexEvent>
    </policy:event>
    <policy:action rdf:resource="#PhotoRemoveRedEyeAction1"/>
  </policy:Policy>
  <policy:SimpleAction rdf:ID="RPhotoAlbumPrintPolicy1Action1C">
    <policy:value>event(RPhotoAlbumPrintPolicy1Event1C0)</policy:value>
  </policy:SimpleAction>
  <policy:SimpleAction rdf:ID="PhotoPolicy1Action1CU0">
    <policy:value>event(RPhotoAlbumPrintPolicy1Event1CU0)</policy:value>
  </policy:SimpleAction>
  <policy:Policy rdf:ID="PhotoAlbumPolicy1C">
    <rdfs:comment>Policy for this service to use removeredeye and create calendar adaptive behaviours
for processing portrait photos</rdfs:comment>
    <policy:target rdf:resource="PhotoAlbumPrint.owl#PhotoAlbum"/>
    <policy:event>
      <policy:SimpleEvent rdf:ID="PhotoAlbumPolicy1CEvent1A">
        <policy:value>RPhotoAlbumPrintPolicy1Event1bA</policy:value>
      </policy:SimpleEvent>
    </policy:event>
    <policy:action>
      <policy:SimpleAction rdf:ID="PhotoAlbumPolicy1CAction1">
        <policy:value>event(PhotoAlbumCreateCalendarEvent)</policy:value>
      </policy:SimpleAction>
    </policy:action>
  </policy:Policy>
  <policy:SimpleAction rdf:ID="RPhotoAlbumPrintPolicy1Action11">
```

```
    <policy:value>event(RPhotoAlbumPrintPolicy1Event1bA)</policy:value>
  </policy:SimpleAction>
  <policy:SimpleAction rdf:ID="RPhotoAlbumPrintPolicy1Action10">
    <policy:value>event(RPhotoAlbumPrintPolicy1Event1aA)</policy:value>
  </policy:SimpleAction>
  <policy:SimpleEvent rdf:ID="RPhotoAlbumPrintPolicy1Event1">
    <policy:value>ProcessEvent</policy:value>
  </policy:SimpleEvent>
  <policy:Policy rdf:ID="RPhotoAlbumPrintPolicy1">
    <rdfs:comment>Policy for this service to use removeredeye and create calendar adaptive behaviours
for processing portrait photos</rdfs:comment>
    <policy:target rdf:resource="PhotoAlbumPrint.owl#PhotoAlbumPrint"/>
    <policy:event rdf:resource="#RPhotoAlbumPrintPolicy1Event1"/>
    <policy:action rdf:resource="#RPhotoAlbumPrintPolicy1Action1C"/>
  </policy:Policy>
  <policy:SimpleCondition rdf:ID="PhotoPolicy1Condition1">
    <policy:subject rdf:resource="#PhotoCategory"/>
    <policy:predicate rdf:resource="#equal"/>
    <policy:value>portrait</policy:value>
  </policy:SimpleCondition>
  <policy:ComplexAction rdf:ID="RPhotoAlbumPrintPolicy1Action1CU">
    <rdfs:first>
      <policy:AndList>
        <rdfs:first rdf:resource="#RPhotoAlbumPrintPolicy1Action10"/>
        <rdfs:rest rdf:resource="#RPhotoAlbumPrintPolicy1Action11"/>
      </policy:AndList>
    </rdfs:first>
    <rdfs:rest></rdfs:rest>
  </policy:ComplexAction>
  <policy:SimpleAction rdf:ID="PhotoPolicy1CAction1">
    <policy:value>event(PhotoRemoveRedEyeEvent)</policy:value>
  </policy:SimpleAction>
  <policy:SimpleEvent rdf:ID="PhotoPolicy1Event1C">
    <policy:value>RPhotoAlbumPrintPolicy1Event1C0</policy:value>
  </policy:SimpleEvent>
  <policy:SimpleEvent rdf:ID="PhotoPolicy1CEvent1A">
    <policy:value>RPhotoAlbumPrintPolicy1Event1aA</policy:value>
  </policy:SimpleEvent>
  <policy:Policy rdf:ID="PhotoPolicy1">
    <rdfs:comment>Policy for this service to use removeredeye and create calendar adaptive behaviours
for processing portrait photos</rdfs:comment>
    <policy:target rdf:resource="PhotoAlbumPrint.owl#Photo"/>
    <policy:event rdf:resource="#PhotoPolicy1Event1C"/>
    <policy:condition rdf:resource="#PhotoPolicy1Condition1"/>
    <policy:action rdf:resource="#PhotoPolicy1Action1CU0"/>
  </policy:Policy>
  <policy:Policy rdf:ID="RPhotoAlbumPrintPolicy1CU">
    <rdfs:comment></rdfs:comment>
    <policy:target rdf:resource="PhotoAlbumPrint.owl#PhotoAlbumPrint"/>
    <policy:event>
      <policy:SimpleEvent rdf:ID="PhotoPolicy1Event1CU">
        <policy:value>RPhotoAlbumPrintPolicy1Event1CU0</policy:value>
      </policy:SimpleEvent>
    </policy:event>
    <policy:action rdf:resource="#RPhotoAlbumPrintPolicy1Action1CU"/>
  </policy:Policy>
  <policy:Policy rdf:ID="PhotoPolicy1C">
    <rdfs:comment>Policy for this service to use removeredeye and create calendar adaptive behaviours
for processing portrait photos</rdfs:comment>
    <policy:target rdf:resource="PhotoAlbumPrint.owl#Photo"/>
    <policy:event rdf:resource="#PhotoPolicy1CEvent1A"/>
    <policy:action rdf:resource="#PhotoPolicy1CAction1"/>
  </policy:Policy>
</rdf:RDF>
```