STATUS OF THESIS

Title of thesis	Achieving Autonomic Service Oriented Architecture Using Case-Based Reasoning

I MUHAMMAD AGNI CATUR BHAKTI

hereby allow my thesis to be placed at the Information Resource Center (IRC) of Universiti Teknologi PETRONAS (UTP) with the following conditions:

1. The thesis becomes the property of UTP
2. The IRC of UTP may make copies of the thesis for academic purposes only.
3. This thesis is classified as

☐ Confidential

☑ Non-confidential

If this thesis is confidential, please state the reason:

_____
_____
_____

The contents of the thesis will remain confidential for _____ years.

Remarks on disclosure:

_____
_____
_____

Endorsed by

_____ _____

Signature of Author | Signature of Supervisor

Permanent address:
Komplek MABAD II No. 40
RT. 02/011 Srengseng Sawah
Jakarta 12640, Indonesia

Assoc. Prof. Dr. Azween B. Abdullah
Computer and Information Sciences Department

Date : _____ Date : _____

UNIVERSITI TEKNOLOGI PETRONAS

ACHIEVING AUTONOMIC SERVICE ORIENTED ARCHITECTURE

USING CASE BASED REASONING

by

MUHAMMAD AGNI CATUR BHAKTI

The undersigned certify that they have read, and recommend to the Postgraduate Studies Programme for acceptance this thesis for the fulfillment of the requirements for the degree stated.

Signature: _____

Main Supervisor: Associate Professor Dr. Azween Bin Abdullah

Signature: _____

Co-Supervisor: Nil

Signature: _____

Head of Department: Dr. Mohd. Fadzil Bin Hassan

Date: _____

ACHIEVING AUTONOMIC SERVICE ORIENTED ARCHITECTURE USING CASE BASED REASONING

by

MUHAMMAD AGNI CATUR BHAKTI

A Thesis

Submitted to the Postgraduate Studies Programme

as a Requirement for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER AND INFORMATION SCIENCES DEPARTMENT

UNIVERSITI TEKNOLOGI PETRONAS

BANDAR SERI ISKANDAR,

PERAK

AUGUST 2011

DECLARATION OF THESIS

Title of thesis	Achieving Autonomic Service Oriented Architecture Using Case-Based Reasoning

I MUHAMMAD AGNI CATUR BHAKTI

hereby declare that the thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTP or other institutions.

Witnessed by

Signature of Author	Signature of Supervisor
Permanent address: Komplek MABAD II No. 40 RT. 02/011 Srengseng Sawah Jakarta 12640, Indonesia	Assoc. Prof. Dr. Azween B. Abdullah Computer and Information Sciences Department
Date :	Date :

ACKNOWLEDGEMENTS

First of all, I would like to say Alhamdulillah, praise Allah The Most Gracious and The Most Merciful for all His blessings and enabling me to finish my PhD. My heartfelt gratitude also goes to my wife and children, my parents, and my family for always supporting me in my personal endeavors. Their relentless encouragement has enabled me to persevere even in troubled times.

I would like to thank my supervisor Associate Professor Dr. Azween Bin Abdullah for his supervision and guidance throughout my study and writing up this thesis. I greatly appreciate his encouragement to my work especially in time of lacking self-confidence, self-discipline, and motivation.

My gratitude is also addressed to Dr. Mohd. Fadzil Bin Hassan for his support as the Head of Department of Computer and Information Sciences and examiner of my thesis, and to the lecturers in the department for their sharing of knowledge, discussion, and reviews.

My sincere thanks go to all the administrative staffs at the CIS Department, Postgraduate Office, and Research Enterprise Office for their assistance during my work and research in UTP. At last but not least, I would also like to express my appreciation to my friends and colleagues: Firman, Hermawan, Petrus, Mr. Totok, and so many more. Thank you for all the experiences, knowledge, and assistance during my stay here at UTP, and hopefully our friendship will not end here.

ABSTRACT

Service-Oriented Architecture (SOA) enables composition of large and complex computational units out of the available atomic services. However, implementation of SOA, for its dynamic nature, could bring about challenges in terms of service discovery, service interaction, service composition, robustness, etc. In the near future, SOA will often need to dynamically re-configuring and re-organizing its topologies of interactions between the web services because of some unpredictable events, such as crashes or network problems, which will cause service unavailability. Complexity and dynamism of the current and future global network system require service architecture that is capable of autonomously changing its structure and functionality to meet dynamic changes in the requirements and environment with little human intervention. This then needs to motivate the research described throughout this thesis.

In this thesis, the idea of introducing autonomy and adapting case-based reasoning into SOA in order to extend the intelligence and capability of SOA is contributed and elaborated. It is conducted by proposing architecture of an autonomic SOA framework based on case-based reasoning and the architectural considerations of autonomic computing paradigm. It is then followed by developing and analyzing formal models of the proposed architecture using Petri Net. The framework is also tested and analyzed through case studies, simulation, and prototype development. The case studies show feasibility to employing case-based reasoning and autonomic computing into SOA domain and the simulation results show believability that it would increase the intelligence, capability, usability and robustness of SOA. It was shown that SOA can be improved to cope with dynamic environment and services unavailability by incorporating case-based reasoning and autonomic computing paradigm to monitor and analyze events and service requests, then to plan and execute the appropriate actions using the knowledge stored in knowledge database.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS, NOMENCLATURE

BPEL	Business Process Execution Language
CBR	Case-Based Reasoning
CDL	Choreography Description Language
CORBA	Common Object Request Broker Architecture
DCOM	Distributed Component Object Model
DDS	Data Distribution Service
HTTP	HyperText Transfer Protocol
QoS	Quality of Service
REST	Representational State Transfer
RPC	Remote Procedure Call
SaaS	Software as a Service
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOC	Service Oriented Computing
SOMF	Service Oriented Modeling Framework
UDDI	Universal Description, Discovery, and Integration
UML	Unified Modeling Language
URI	Uniform Resource Identifier
WCF	Windows Communication Foundation
WSDL	Web Services Description Language
XML	eXtensible Markup Language

CHAPTER 1

INTRODUCTION

This chapter presents an introduction to the conducted research covering an overview of service-oriented computing (SOC) and service-oriented architecture (SOA). Thereafter, an overview of the issues and problems in SOA and the objectives of the research are given. An outline of the remaining chapters of this thesis will be the last part of this chapter.

1.1 Introduction

As the development of internet and World Wide Web technologies has enabled an access to many types of services over the web, networked and distributed systems (providing resources, services, etc.) are nowadays gaining an increasing importance and demand. Hence, the scale and complexity of current distributed systems are also increasing and showing high dynamism [Montresor et al., 2002]. Furthermore, on the base of existing services, large distributed computational units can be built by composing complex compound services out of simple atomic ones [Lazovik & Arbab, 2007]. This type of concept and architecture is called Service-Oriented Computing (SOC) and Service-Oriented Architecture (SOA) respectively.

Service-oriented computing is an emerging computing paradigm that utilizes services as the basic constructs to support the development of rapid and easy composition of distributed applications. The visionary promise of SOC is to assemble the application components with little effort into network of services that can be loosely coupled and used to create the flexible dynamic business processes and applications that may span organizational boundaries and computing platforms.

Fig. 1.1 shows an example of a service-oriented model whose components (data, software, platforms, etc.) should be considered as service that can be used by users through the network, despite of the underlying technologies being used to provide those services.

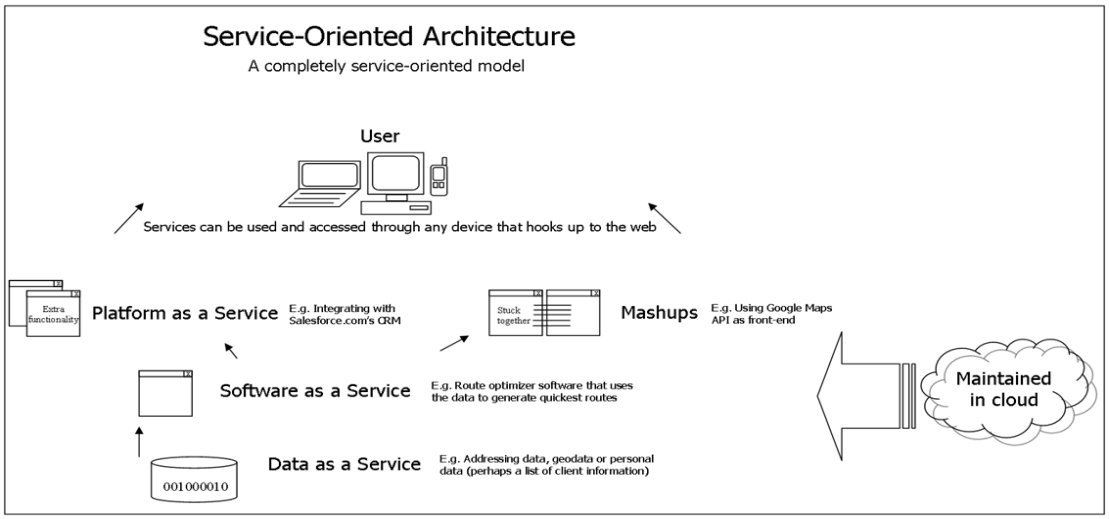

Fig. 1.1 A service oriented model

Fig. 1.2 shows a small scale SOA as an illustrative example. A single business process engine is deployed using service-based integration adapters to access a services based message broker (e.g., the one offered by an enterprise service bus). Service-based business application adapters are used to access several back-end systems, such as databases or legacy systems. The service adapter interface is hence used to unify the interfaces to different kinds of the back-end systems. A typical SOA in organizations today is much larger than this illustrative example in that multiple process engines – e.g., one per department – are deployed, plus multiple instances of all other components. These are again integrated using the same service-oriented interface, i.e., the business process engines that can invoke business processes in other engines as sub-processes via the service-oriented invocation interface of the engine.

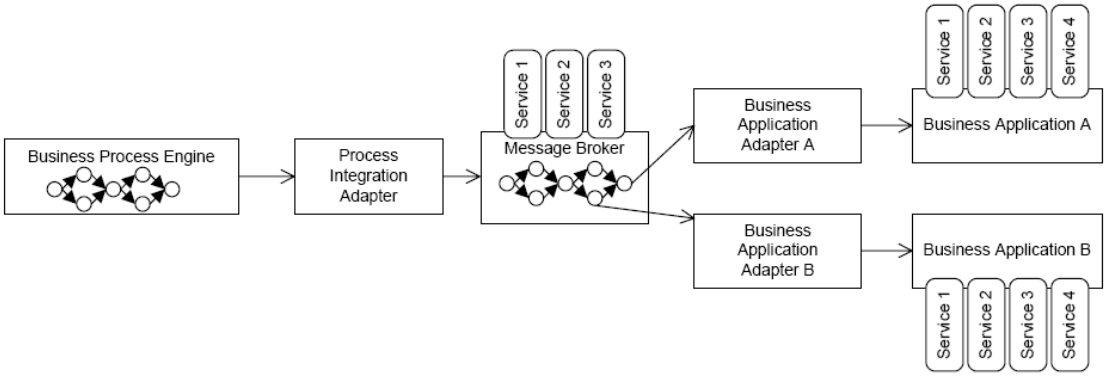

Fig. 1.2 Example of a small scale SOA

Fig. 1.3 illustrates a comparison of traditional software architecture and SOA. In traditional architecture, the software system is static in which any changes, updates, patch, plug-ins, etc need to be maintained on site. On the other hand, SOA positions every component of the system (data, software, and platform) as services maintained in the network cloud.

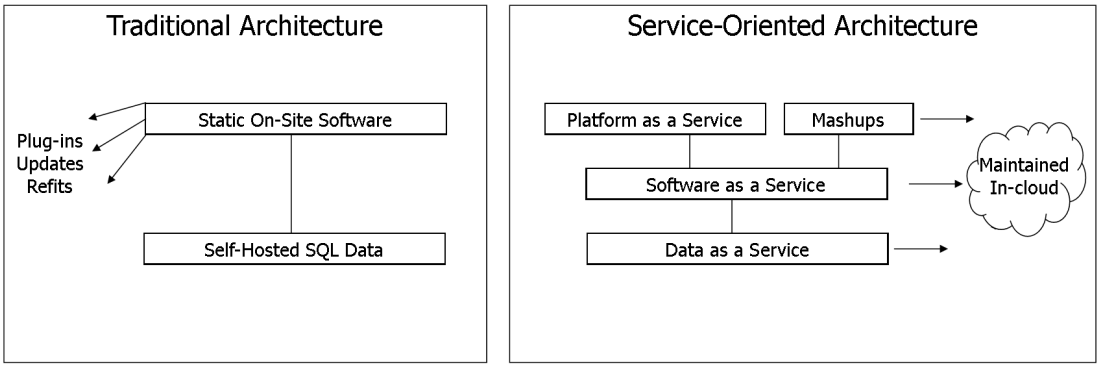

Fig. 1.3 Comparison of traditional software architecture and SOA

The subject of SOA is vast and complex, spanning many concepts and technologies that find their origins in some diverse disciplines that are intricately woven together. In addition, it needs to merge technology with an understanding of business processes and organizational structures, a combination of recognizing an enterprise's problems and the potential solutions that can be applied to make them correct. Many challenges and issues accordingly arise on the subject.

Due to its dynamic nature, implementation of SOA might emerge challenges which include service discovery, service interaction, service composition, robustness, Quality of Service (QoS), security, etc. SOA may often to dynamically organizing and re-organizing its topologies of interactions among the services. Furthermore, complexity and dynamism of the current global network system require architecture to be capable of autonomously changing its structure and functionality to meet the dynamic changes in the requirements and environment without involving much human intervention.

1.2 Problem Statement

Although some techniques have been proposed to address the issues present in SOC and SOA, these issues, due to the complexity of SOC and SOA technologies, still exist and remain as open and active research fields [Arbab, 2008]. In terms of service discovery, the issues and questions including the following: how to discover the really needed services and how to interact with the discovered services. In term of service composition, some of the questions include the following: how to adapt to incongruent (non-similar) services, how to elaborate to build a new service, and how to ensure that the composed services work properly, i.e., robust, and secure with an adequate quality of service.

The current SOA frameworks offer agility, maintainability, reusability, consistency, efficiency, integration and reduced cost of a service [Bell, 2008], [Rosen et al., 2008], [Schneider et al., 2008]. Yet, they are still lacking for adaptability and robustness. Schneider et al., (2008) stated that technologies and methods are still needed for development of adaptive SOA systems. The result in [Yoo et al., 2009] showed that typical service composition will be complete and correct with an assumption that there are no exceptions or errors occurred from the initiating user to the terminating one. However that is not the case with the current and future complex and dynamic systems.

The work in [Montresor et al. 2002] reported that the scale and complexity of current distributed systems are increasing and showing high dynamism in that the global network systems grow. Future systems also need to be able to cope with unpredictable events that could cause services unavailability, such as crashes or network problems. Therefore, a more robust, more adaptive and autonomous service architecture that can keep up with the dynamic changes in environments and requirements to some extent is required.

In the past years, biological and nature inspired approaches have been proposed as a strategy to handle several complex computer systems. The goal is to obtain some methods in engineering the systems, which have similar high stability and robustness that are frequently found in biological entities. Two of the mechanisms adapted from nature into the computer systems are autonomic computing paradigm and self-organizing systems.

Autonomic computing paradigm [Kephart & Chess, 2003] was inspired from the autonomic nervous system in human and has been proposed to achieve autonomic and self-managing computer systems. Self-organizing systems meanwhile have been discovered in nature and may offer the computational systems that are robust, secure, self-organizing, and self-healing [Hart et al., 2007].

A required dynamic characteristic of future SOA is quite similar with a characteristic of autonomic and self-organizing system, that is on how they are able to organize elements (services in the case of SOA) to change their functions or create new functions on higher levels (emergence). As SOA will need to have some characteristics of autonomic and self-organizing systems, i.e., dynamism, flexibility, adaptivity, it is visionary promising to adapt biological or nature inspired mechanisms into service-oriented computing to creating a more robust, intelligent, and autonomic SOA.

Nevertheless it still remains several problems of how to successfully adapt autonomic and self-organizing mechanisms from nature into SOA. Some following questions then arise on the subject, such as: which nature / biological inspired process

are promising to be adapted into SOA? How to adapt the nature / biological inspired process into SOA? What and how much benefits can be gained by adapting that nature process into SOA?

Based on the aforementioned issues and problems, some research questions could be derived as follows:

- How to adapt autonomic models into SOC / SOA? First, it is necessary to do an initial literature review, to identify a process / model / mechanism / paradigm having autonomic and self-organizing characteristics that is potentially suitable for SOA and then to map that model into SOA domain.
- What are the specifications of a service oriented architectural framework that will ensure adaptive and autonomic SOC / SOA?
- What are the required components (software, services, etc) needed to support the architectural framework development?
- Would the adaptation of nature-inspired mechanisms into SOA improve its quality or performance? To do this, it is essential to identify the metrics and the tools or techniques to measure the improvements.

1.3 Objectives

The main objective of the research is to extend the capability and intelligence of service oriented architecture by adapting autonomy into service-oriented computing that can be used to develop more robust, intelligent, and autonomic service oriented framework. To achieve this, a number of specific goals have been defined as follows:

- To design and develop a more adaptive, intelligent, and autonomic SOA framework based on the concepts of adapting self-organization / self-configuration into service-oriented architecture. The output of this objective is the autonomic SOA framework that will answer the first and second research questions by providing the design and specifications of the framework.

- To develop models and simulation or prototype of the proposed autonomous service-oriented architecture as proof of concept. The results of the modeling and simulation / prototype development will be analyzed and compared against the results from other researches. The outputs of this objective are the models and simulation or prototype of the proposed framework and their results. These will answer the third and forth research questions by providing the simulation / prototype development specifications and its quality measurements.

1.4 Impact and Contributions

The research will bring a significant positive impact on different areas in service-oriented computing, including addressing new open research issues on how services are composed and maintained. The main achievement from this research will be the design and development of an autonomic SOA framework that is robust, intelligent, and autonomous (flexible, adaptable, and resilient). The proposed framework consists of a set of models within service-oriented architecture.

The major impact of the research spans over the following areas:

- New service architecture inspired by an autonomic computing paradigm that autonomously adapt and organize its services interactions.
- New autonomic mechanism in SOA that autonomously monitors and analyzes service requests, as well as plans and provides the optimum services.
- New adaptive and self-learning mechanism in SOA that remembers service profiles leading to better and faster reactions in the future, utilizing case-based reasoning.
- Less human intervention is required for service discovery and compositions during operation (autonomous).
- Formal (Petri Net based) and UML based models of the proposed framework.
- Comprehensive overall service architecture for various service ecosystems as overviewed in chapter five, describing some available real-world applications that will benefit from the proposed work.

- Extendable service architecture towards digital service ecosystems using collaboration agreement.

1.5 Research Methodology

The methodology used in this research includes:

1. Literature study on service-oriented architecture, autonomic computing paradigm, self-organizing systems, case-based reasoning, and other related subjects.
2. Concept and theory formulation of autonomic and self-organizing service-oriented architecture.
3. Architectural framework design and development of the autonomic service oriented architecture.
4. Evaluation through modeling and case studies.
5. Simulation and prototype development.
6. Discussion and analysis of the simulation and prototype development results.

The detail of the methodology will be discussed further in chapter three.

1.6 Scope of Work

It is the aim of this research to extend the capability and intelligence of service oriented architecture within the constraint of autonomic computing paradigm. There are four sub-concepts within autonomic computing paradigm; they are self-configuration (self-organize), self-optimization, self-healing, and self-protection [Kephart & Chess, 2003]. This research is concerned with how existing services available on the network can be integrated within a more adaptive service oriented architecture that can adapt, configure and organize its services interactions autonomously. For this purpose, this research particularly focuses only on the self-organizing / self-configuring concept of autonomic computing.

There are ongoing research initiatives within the broad field of SOA attempting to enhance its capability and intelligence. For example, [Erl, 2007] introduces service autonomy to support the extent to which software design principles can be effectively realized in real world environments, and [Schneider et al., 2008] gave an overview of on adaptive service-based systems based on software engineering approaches. Those researches tend to employ pure software engineering approaches to provide a form of intelligence in SOA. On the other hand, this research employes software engineering techniques inspired from human's systems, namely autonomic computing paradigm inspired from human's autonomic nervous system [Kephart & Chess, 2003], and case-based reasoning inspired from human's problem solving [Kolodner, 1992], [Aamodt & Plaza, 1994]. Therefore the approach employed in this research is a hybrid, combining biological inspired models (human's system in this case) and software engineering approach, aiming to harness both the predictability behavior of software engineering approach and the adaptability of biological systems.

1.7 Thesis Structure

This thesis is divided into seven chapters. Chapter one introduces the background that comprises the reasons of conducting the research, the problems and the approach used to solve the problem. It also describes the objectives, expected impact and contributions of this research.

Chapter two elaborates several comprehensive and extensive reviews of enabling technologies used to address the research problems and the current solution. This chapter also provides literature review and discussion of related research works. Chapter three, from the issues highlighted in chapter one and two, presents the methodology and techniques used in this research.

Chapter four derives and discusses the models and formal representation of the proposed framework. Then chapter five discusses the possible application domain and case studies where the proposed framework can be applied and tested through several case studies chosen.

Chapter six further discusses about the prototype design and development, issues and experience gained from the development process during the research. This chapter then also discusses and analyzes on the development and implementation results. Finally, chapter seven presents the conclusions of the research and recommendation for future works.

CHAPTER 2

LITERATURE REVIEW

2.0 Chapter Overview

This chapter presents a background review on service-oriented computing and architecture, autonomic computing paradigm, self organizing systems, case-based reasoning, and elaboration on some selected works related to this research.

2.1 Service Oriented Computing and Service Oriented Architecture

Service-oriented computing (SOC) is an emerging computing paradigm that utilizes services as the basic constructs to support the development of rapid and easy composition of distributed applications – even in a heterogeneous environment. Service-Oriented Architecture (SOA) meanwhile is a main architectural concept in the field of SOC. In this kind of architecture, all functions, or services are defined by using a description language and have platform independent interfaces that could be invoked and called to perform business processes. Each service is an end point of a connection, which can be used to access the service, and each interaction is relatively independent of each and every other interaction.

Service-oriented architecture refers to a method for a development and integration of a system in which functionality is grouped around business processes and packaged as the interoperable services. It is a design for linking computational resources (principally applications and data) on demand to achieve the desired results for service consumers (the end users or other services). OASIS (The Organization for the Advancement of Structured Information Standards) defines SOA as follows:

"*A paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations*" [OASIS Reference Model for SOA, 2006]. The World Wide Web Consortium (W3C) defines SOA as the following: *"A set of components which can be invoked, and whose interface descriptions can be published and discovered"* [Haas & Brown, 2004].

Service-oriented architecture also describes an IT infrastructure wherein different applications that participate in business processes exchange data with one another. The aim is to loosen the coupling of services through operating systems, programming languages, and other technologies underlying the applications. SOA separates functions into distinct units, or services to be accessible over a network to make them combinable and reusable in the production of business applications. These services communicate with each other by passing data from one service to another, or by coordinating an activity between two or more services. SOA concepts are often seen to be built upon and evolved from the former concepts of distributed computing and modular programming (component-based software engineering).

The service composition layer is typically on top of the various layers of functionality that implement a SOA and provides a process engine (or workflow engine), which invokes the SOA services to realize individual activities in the process. The main goal of such process-driven SOA is to increase productivity, efficiency, and flexibility of an organization via process management. This is achieved by aligning the high-level business processes with the applications supported by IT. Changes in business requirements are carried out as changes in the high-level business processes implemented by linking their new activities to existing or new IT-supported applications. Organizational flexibility can be achieved because the explicit business process models are easier to change and evolve than, for instance, the hard-coded business processes in the program code. For the long term, the goal is to enable a business process improvement through IT.

According to [He, 2003] there are four rules to follow before architecture can be considered to be service-oriented:

1. The service messages must be descriptive, rather than instructive for the service provider is responsible for solving the problem.

2. Service providers will be unable to understand the request if the messages are not written in an understandable format, structure, and vocabulary. Limiting the vocabulary and structure of messages afterward is a necessity for any efficient communication. The more restricted a message is, the easier it is to understand, although it comes at the expense of reduced extensibility.

3. Extensibility is vital. Same as the real world, any environment in which a software system lives is an ever-changing place. Those changes in turn demand corresponding changes in the software system, service consumers, providers, and the messages they exchange. If messages are not extensible, consumers and providers will be locked into one particular version of a service. Despite the importance of extensibility, it has been traditionally overlooked. At best, it was regarded simply as a good practice rather than something fundamental. Restriction and extensibility are deeply entwined and equally needed. As increasing one might come at the expense of reducing the other, a right balance is needed.

4. SOA must have a mechanism enabling a consumer to discover a service provider under the context of a service sought by the consumer. The mechanism can be really flexible, and does not have to be a centralized registry.

SOA implementations rely on a mesh of software services comprising the unassociated, loosely coupled units of functionality that have no calls to each other embedded in them. Each service implements only one action, such as filling out an online application for an account, viewing an online bank statement, or placing an online booking or airline ticket order. Rather than services embedding calls to each

other in their source code, they use some defined protocols that describe how services pass and parse messages using description metadata.

SOA developers associate individual SOA objects by using orchestration. Here, the developer associates software functionality (the services) in a non-hierarchical arrangement using a software tool that contains a complete list of all available services, characteristics, and the means to build an application utilizing these sources.

Underlying and enabling all of this require metadata in sufficient detail to describe not only the characteristics of these services, but also the data that drives them. Programmers for example have made an extensive use of XML (Extensible Markup Language) in SOA to structure data wrapped in a nearly exhaustive description-container. Analogously, the Web Services Description Language (WSDL) [Christensen et al, 2001], [Chinnici et al, 2007] typically describes the services themselves, while the SOAP protocol describes the communications protocols. Whether these description languages are the best possible for the job or will become / remain the favorites in the future, still remains open questions.

SOA depends on data and services described by metadata that should meet the following two criteria:

1. The metadata should be in a form in which the software systems can use to configure dynamically by discovery and incorporation of defined services, and also to maintain coherence and integrity. For example, it could be used by other applications to perform discovery of services without modifying the functional contract of a service.

2. The metadata should be in a form in which the system designers can understand and manage with a reasonable expenditure of cost and effort.

SOA aims to allow users to simultaneously string fairly large chunks of functionality to form ad hoc applications that are built almost entirely from existing software services. The larger the chunks are, the fewer the interface points are

required to implement any given set of functionality. However, very large chunks of functionality may not prove sufficiently granular for easy reuse. Each interface brings with it amount of processing overhead, so there is a performance consideration in choosing the granularity of services. The great promise of SOA suggests that the marginal cost of creating the n-*th* application is low, as all of the software required already exists to satisfy the requirements of other applications. Ideally, one requires only orchestration to produce a new application.

To make it works, there should be no interactions between the specified chunks or within the chunks themselves. Instead, human specifies the interaction of services (all of them unassociated peers) in a relatively ad hoc way with an intent driven by newly emergent requirements. Programmers develop the services by themselves using traditional languages such as Java, C, C++, C#, Visual Basic, COBOL, or PHP.

SOA services feature loose coupling, in contrast to the functions that a linker binds together to form an execution to a dynamically linked library or to an assembly. SOA services also run in wrappers (such as Java or .NET) and in other programming languages that manage memory allocation and reclamation, allow ad hoc and late binding, and provide some degree of indeterminate data typing.

Increasing numbers of third-party software companies have offered software services for a fee. In the future, SOA systems might consist of such third-party services combined with others created in-house. It then potentially will spread costs over many customers and customer uses, and promote standardization both in and across industries. In particular, the travel industry now has a well-defined and documented set of both services and data, sufficient to allow any reasonably competent software engineer to create travel-agency software using the entirely off-the-shelf software services. Other industries, such as finance industry, additionally have started making a significant progress through this direction.

SOA as an architecture relies on a service-orientation as its fundamental design principle. If a service presents a simple interface that abstracts away its underlying

complexity, users might access independent services without knowledge of the service's platform implementation.

Enterprise software architects believe that SOA could assist businesses to respond more quickly and cost-effectively to the changing market-conditions. This style of architecture promotes reuse at the macro (service) level rather than micro (classes) one. It additionally can simplify an interconnection to – and a usage of – existing IT (legacy) assets.

In some respects, SOA could be regarded as an architectural evolution rather than as a revolution, meaning that it captures many of the best practices of some previous software architectures. In communications, for example, little development has taken place of solutions that use the truly static bindings to talk to other equipments in network. By formally embracing a SOA approach, such systems can position themselves to stress the importance of well-defined, highly inter-operable interfaces. SOA further promotes the goal of separating users (consumers) from the service implementations. Services can therefore be run on various distributed platforms and be accessed across networks as well as maximize reuse of services.

When creating services, SOA realizes its business and IT benefits by utilizing an analysis and design methodology. This methodology ensures that services remain consistent with the architectural vision and roadmap, and adhere to the principles of service-orientation.

A service comprises a stand-alone unit of functionality available only via a formally defined interface. Services can be some kind of "nano-enterprises" that are easy to produce and improve, i.e., *atomic services*. It can also be "mega-corporations" constructed as a coordinated work of sub-ordinate services, i.e., *composite services*. [Bell, 2008] included the third entity in SOA asset which is the *service cluster*, i.e., group of entities based on affiliation, relationship, and business or technology context.

Reasons for treating the implementation of services as the separate projects from larger projects include:

1. Separation promotes a concept to a business that services can be quickly and independently delivered from the larger and slower-moving projects commonly in the organization. The business starts understanding systems and simplified user interfaces calling on services. This advocates agility fostering business innovations and speeds up time-to-market.

2. Separation promotes the decoupling of services from consuming projects. This encourages a good design insofar as the service is designed without knowing who its consumers are.

3. Documentation and test artifacts of the service are not embedded within the detail of the larger project. This is important when the service needs to be reused later. Examples may prove useful to aid in documenting a service to the level where it becomes useful. The documentation of some APIs within the Java Community Process provides good examples. As these are exhaustive, users would typically use only important subsets.

If an organization possesses an appropriately defined test data, a corresponding stub will be built to react to the test data when a service is being built. A full set of regression tests, scripts, data, and responses is also captured for the service. The service can be tested as a black box using the existing stubs corresponding to the services it calls. Test environments can be constructed where the primitive and out-of-scope services are stubs, while the remainders of the mesh are test deployments of full services. As each interface is fully documented with its own full set of regression test documentation, it becomes simple to identify problems in test services. Testing evolves to merely validate that the test service operates according to its documentation, and finds gaps in documentation and test cases of all services within the environment. Managing the data state of idempotent services is the only complexity.

Software designers can implement SOA using a wide range of technologies, including:

- SOAP, RPC (Remote Procedure Call)
- REST (Representational State Transfer)
- DCOM (Distributed Component Object Model)
- CORBA (Common Object Request Broker Architecture)
- Web Services
- DDS (Data Distribution Service)
- WCF (Windows Communication Foundation)

Implementations can use one or more of these protocols and, for example, might use a file-system mechanism to communicate data conforming to a defined interface-specification among processes conforming to the SOA concept. The key is the independent services with the defined interfaces that can be called to standardly perform their tasks, without a service having foreknowledge of the calling application, and without an application having or needing knowledge of the actual task performance of the service.

SOA enables the development of applications built by combining loosely coupled and interoperable services. These services inter-operate based on a formal definition (or contract, e.g., WSDL) that is independent of the underlying platform and programming language. The interface definition hides the implementation of the language-specific service. SOA-based systems can therefore independently function for development technologies and platforms (such as Java, .NET, etc). Services written both in C# running on NET platforms and in Java running on Java EE platforms for example can be consumed by a common composite application (or client). Applications running on either platform can also consume services running on the other as web services that facilitate reuse. Managed environments can furthermore wrap COBOL legacy systems and present them as software services. This has extended the useful life of many core legacy systems indefinitely, no matter what language is originally used.

High-level languages such as Business Process Execution Language (BPEL) and specifications such as Web Services-Choreography Description Language (WS-CDL) and WS-Coordination extend the service concept by providing a method of defining

and supporting orchestration of fine-grained services into more coarse-grained business services in which architects can in turn incorporate into workflows and business processes implemented in the composite applications or portals.

2.1.1 SOA Model

Service oriented architecture modeling is as a framework of SOA that identifies various disciplines and as guidance for SOA practitioners to conceptualize, analyze, design, and architect their service-oriented assets. Such frameworks including UML meta-model [Zhang et al., 2006] and Service Oriented Modeling Framework (SOMF) [Bell, 2008] offers a modeling language and a work structure or "map" depicting various components that contribute to a successful service-oriented modeling approach. It further illustrates the major elements that identify the “what to do” aspects of a service development scheme. The model enables practitioners to craft a project plan and to identify the milestones of a service-oriented initiative. Moreover, SOMF provides a common modeling notation to address an alignment between business and IT organizations. Fig. 2.1 shows the Service Oriented Modeling Framework version 2.0.

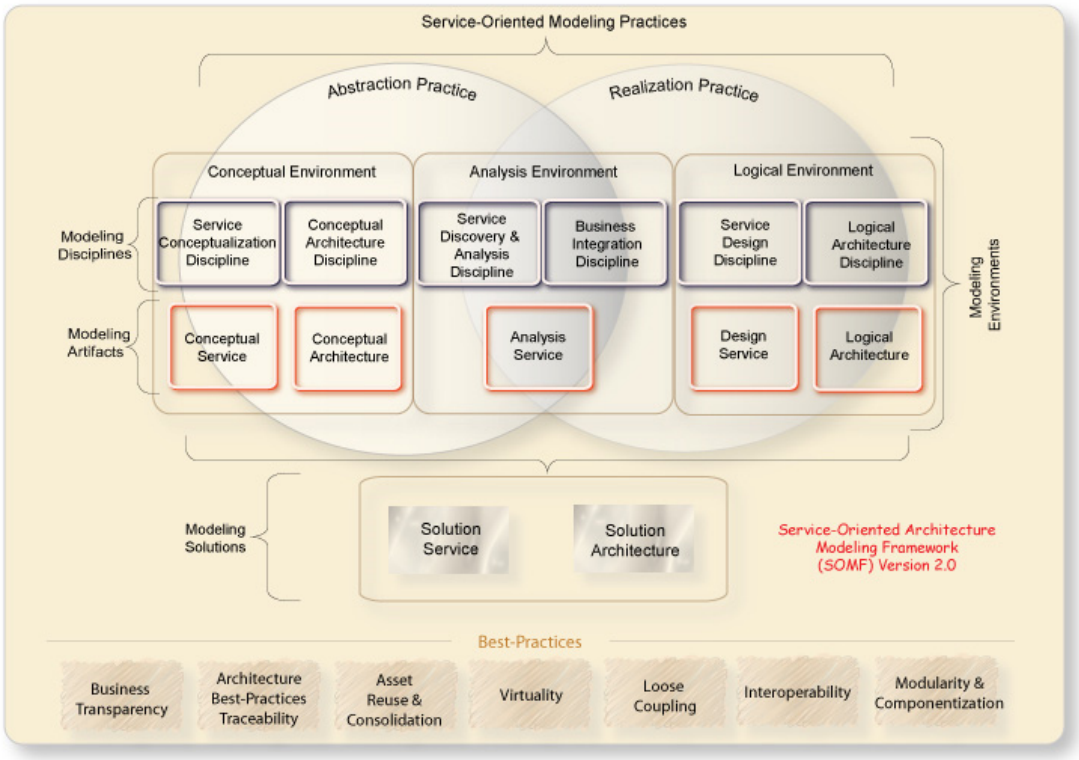

Fig. 2.1 Service-Oriented Modeling Framework (SOMF) [Bell, 2008]

2.1.2 Web Services

Web services, the most common way to implement a SOA, refer to accessing services over the web [Heydarnoori et al, 2006] although there is no definition of web service universally accepted currently. As defined by The W3C (World Wide Web Consortium) Web Services Architecture Working Group, Web service refers to a software system designed to support an interoperable machine-to-machine interaction over a network [Haas & Brown, 2004]. Its major focus is to make the functional building blocks accessible over standard Internet protocols that are independent from platforms and programming languages. These services could be new applications or merely wrapping around the existing legacy systems that make them network-enabled.

Despite the difficulty of defining web services, it is generally accepted that a web service is a SOA with at least the following additional constraints:

- Interfaces must be based on Internet protocols such as HTTP, FTP, and SMTP.
- Except for binary data attachment, messages must be in XML (Extensible Markup Language).

There are two main styles of Web services, namely:

- SOAP (Simple Object Access Protocol) web services
- REST (Representational State Transfer) web services.

A SOAP web service introduces the following constraints:

- Messages, except for binary data attachment, must be carried by SOAP.
- A description of a service must be in WSDL (Web Service Description Language).

A SOAP web service is the most common and marketed form of web service in industry, and most people simply put the term “web service” into SOAP and WSDL services. SOAP provides a message construct that can be exchanged over a variety of underlying protocols according to the SOAP 1.2 Primer. In other words, SOAP acts like an envelope that carries its contents. One advantage of SOAP is that it allows rich message exchange patterns ranging from traditional request-and-response to broadcasting and sophisticated message correlations. There are two types of SOAP web services: SOAP RPC (Remote Procedure Calls), which are not SOA and document-centric SOAP, which conversely are SOA.

SOAP RPC web service breaks the constraint required by an SOA and encodes RPC in SOAP messages. In other words, it "tunnels" new application-specific RPC interfaces through an underlying generic interface. Effectively, it prescribes both system behaviors and application semantics. Since the system behaviors are very difficult to prescribe in a distributed environment, applications created with SOAP RPC are not interoperable by nature. Many real life implementations have confirmed this phenomenon [Zwicky et al., 2000], [He, 2003], [Hu, 2006], [Lynch, 2007].

Faced with this difficulty, both WS-I basic profile and SOAP 1.2 have made the support of RPC optional even though SOAP was originally designed just for RPC. RPC also tends to be instructive rather than descriptive, which is against the spirit of SOA. It will not be a surprise when someone thinks that "SOAP" actually stands for "SOA Protocol".

Meanwhile, the term REST was first introduced by [Fielding, 2000] to describe the web architecture. A REST web service is an SOA based on the concept of "resource", anything that has a Uniform Resource Identifier (URI). A resource, which may have zero or more representations, commonly is considered to be absent if representation is unavailable for that resource. A REST web service requires the following constraints:

1. Interfaces are limited to HTTP. The following semantics are defined:
   - HTTP GET is used to obtain a representation of a resource. A consumer uses it to retrieve a representation from a URI. Services provided through this interface must not incur any obligation from consumers.
   - HTTP DELETE is used to remove representations of a resource.
   - HTTP POST is used to update or create the representations of a resource, and
   - HTTP PUT is used to create representations of a resource.

2. Most messages are in XML, confined by a schema written in a schema language such as XML Schema from W3C or RELAX NG.

3. Simple messages can be encoded with URL encoding.

4. Service and service providers must be resources while a consumer can be a resource.

REST web services require little infrastructure support apart from standard HTTP and XML processing technologies, which are now well supported by most programming languages and platforms. Here, REST web services are simple and

effective as HTTP is the most widely available interface, and quite good for most applications. In many cases, the simplicity of HTTP simply outweighs the complexity of introducing an additional transport layer.

Fig. 2.2 shows web services architectural model given by [Huhns & Singh, 2005]. As a basis for SOA, web services models incorporate how web services are advertised, discovered, selected, and used. The architecture in turn has three main parts:

1. Service provider
2. Service consumer (requestor)
3. Service registry

Providers publish or announce their services on registries using Web Service Description Language (WSDL), where consumers find – using Universal Description Discovery and Integration (UDDI) [OASIS Standard, 2003], [OASIS Standard, 2005], and then invoke them.

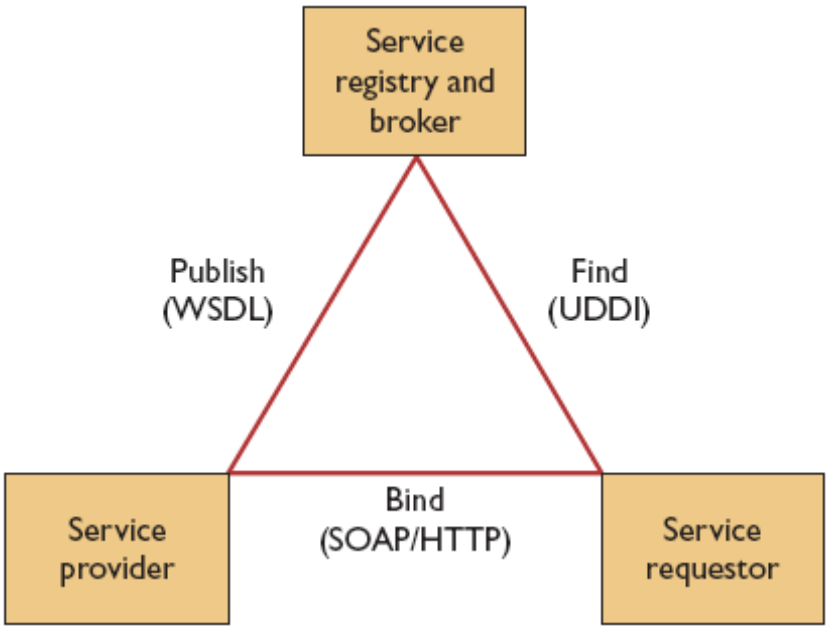

Fig. 2.2 Web service architectural model [Huhns & Singh, 2005]

Fig. 2.3 illustrates a meta-model showing the relationships among the three parties, as given by [Arsanjani, 2005].

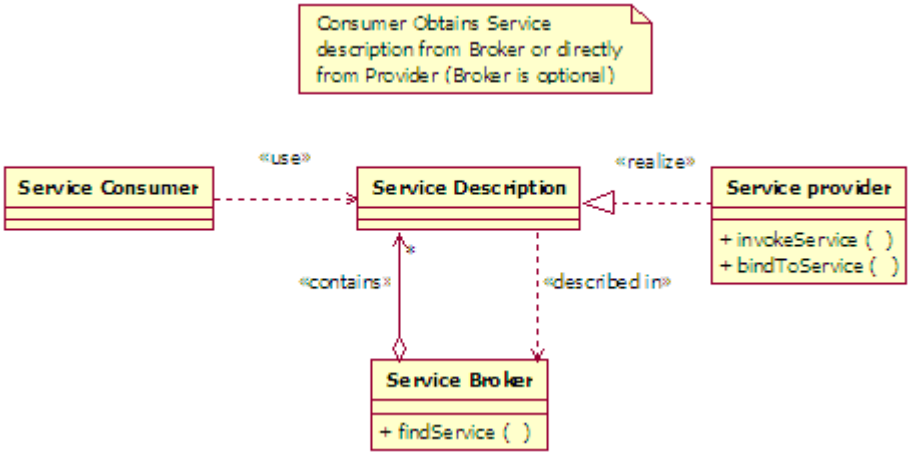

Fig. 2.3 SOA meta-model [Arsanjani, 2005]

The two major roles in SOA are described below, while the service registry is described in the next section.

1. Service provider:

   A service provider creates a web service, possibly publishes its interface and access information to the service registry. Each provider must decide which services to expose, how to make several trade-offs between security and easiness of availability, how to price the services, or (if no charges apply) how/whether to exploit them for other values. The provider additionally has to decide the service category that should be listed in for a given broker service and sorts of trading partner agreements required to use the service. It registers the services available within it, and lists all potential service recipients. The implementer of the broker then decides the scopes of the broker; those are public brokers that are available through the Internet, and private brokers that are only accessible to a limited audience, for example, users of a company intranet. Furthermore, the amount of the offered information has to be decided in which some brokers might specialize in many listings, and others might offer high levels of trust in the listed services or some cover a broad landscape

of services and others might focus within an industry as well as catalog other brokers.

Depending on the business model, brokers can attempt to maximize look-up requests, number of listings or accuracy of the listings. The Universal Description Discovery and Integration (UDDI) specification defines a way to publish and discover information about web services. Other service broker technologies include ebXML (Electronic Business using eXtensible Markup Language) and those based on the ISO/IEC 11179 Metadata Registry (MDR) standard.

2. Service consumer:
   The service consumer or web service client locates entries in the broker registry using various find operations and then binds to the service provider in order to invoke one of its web services. Whichever service the consumers need, they have to take it into the brokers, and then bind it with respective service before using. They can access multiple services if the service provides multiple services.

2.1.3 Service Registry

Service registry containing access information to the applications, i.e. the services they provide is a repository of service and data descriptions, which may be used by service providers (applications) to publish their services, and service requestors (users) to discover available services.

The access information are stored as Web Services Description Language (WSDL) documents describing the protocol bindings and message formats required to interact with the web services listed in its directory. According to WSDL version 1.1, the objects in WSDL documents include the followings:

- Service: The service can be thought of as a container for a set of system functions that have been exposed to the web based protocols.

- Port: The port does nothing more than defining the address or connection point to a web service. It is typically represented by a simple HTTP URL string.

- Binding: Specifies the port type, defines the SOAP binding style (RPC/Document) and transport (SOAP Protocol). The binding section also defines the operations.

- Port Type: The <portType> element defines a web service, the operations that can be performed, and the messages used to perform the operation.

- Operation: Each operation can be compared to a method or function call in a traditional programming language. Here the SOAP actions are defined and the way the message is encoded for example, "literal."

- Message: Typically, a message corresponds to an operation and contains the information needed to perform the operation. Each message consists of one or more logical parts, which of each is associated with a message-typing attribute. The message name attribute provides a unique name among all messages, and provides a unique name among all the parts of the enclosing message. Parts are a description of the logical content of a message. In Remote Procedure Call (RPC) binding, a binding may refer to the name of a part in order to specify binding-specific information about the part. A part may represent a parameter in the message, while the bindings define the actual meaning of the part.

- Element: Elements are defined within the <types> tag. An element consists of a unique name, and data type. The purpose of an element WSDL is to describe the data and to define the tag which delimits the data sent in the message parameters. The elements can be simple types (such as strings or integers) that can have enumerations (lists of acceptable values) or restrictions defined

(length not to exceed 10 characters). In addition, they can have complex types that can nest other elements with in them.

- XSD Files: Elements are often defined in an XML Schema Definition (XSD) file. The XSD can be in the same WSDL file or in a separate file. It is imported to the WSDL through the use of the WSDL import tag with a reference to the namespace of the XSD document. When an XSD refers to elements defined in another XSD file, the external XSD namespace must be imported into the XSD referencing to the element. If the XSD is not defined directly in the WSDL, the namespace specifies the location of the XSD file in URL syntax.

2.2 Autonomic Computing Paradigm

In computer science, self-management is the main concepts in autonomic computing paradigm [IBM, 2001], [Kephart & Chess, 2003], which consists of the sub-concepts in autonomic computing, e.g. self-configuration, self-optimization, self-healing, and self-protection [Kephart & Chess, 2003]. Autonomic computing paradigm has been proposed as an approach for the development of applications of computer and software systems that can manage themselves given only high-level objectives from human. It also has been used in many researches in various domains such as those in [White et al., 2004], [Parashar & Hariri, 2004], [Arora et al., 2006], [Wang, 2007], [Montani & Anglano, 2008] and continuously being studied and researched.

Autonomic computing paradigm was introduced by IBM's senior vice president, Paul Horn, in 2001 [IBM, 2001], inspired by the autonomic nervous system that govern some functions in human body, such as heart rate and body temperature, and freeing human's conscious brain from the burden of dealing with these and many other vital functions.

According to [Kephart & Chess, 2003], an autonomic element will typically consist of one or more managed elements coupled with a single autonomic manager that controls and represents them. The managed element will essentially be equivalent

to what is found in ordinary non-autonomic systems, although it can be adapted to enable the autonomic manager to monitor and control it. The managed element could be a hardware resource, such as storage, CPU, printer, or a software resource, such as a database, directory service, or large legacy system.

At the highest level, the managed element could be an e-utility, an application service, or even an individual business. The autonomic manager distinguishes the autonomic element from its non-autonomic counterpart. By monitoring the managed element and its external environment, constructing and executing plans based on an analysis of this information, the autonomic manager will relieve humans of the responsibility of directly managing the managed element.

The autonomic computing paradigm has changed the view of the fundamental definition of the technology age, from one of computing to one defined by data [IBM, 2001]. These systems after applying autonomic computing paradigm to computer systems, software, and storage will have the following properties:

- Flexible: the system will be able to examine data via an agnostic approach.
- Accessible: the nature of autonomic system is always accessible.
- Seamless: the system will perform its tasks and adapt to the user's needs without involving the user into its work intricacies.

Fig. 2.4 shows the structure of an autonomic element. The autonomic systems consist of autonomic elements, whose behavior is controlled by autonomic manager, which shall relieve the human responsibility of directly managing the managed elements by monitoring these elements and its external environment to construct and execute plans based on the analysis of the gathered information. That is, the autonomic managers will carry out the autonomic computing cycle: monitoring, analyzing, planning, and executing, via its knowledge base.

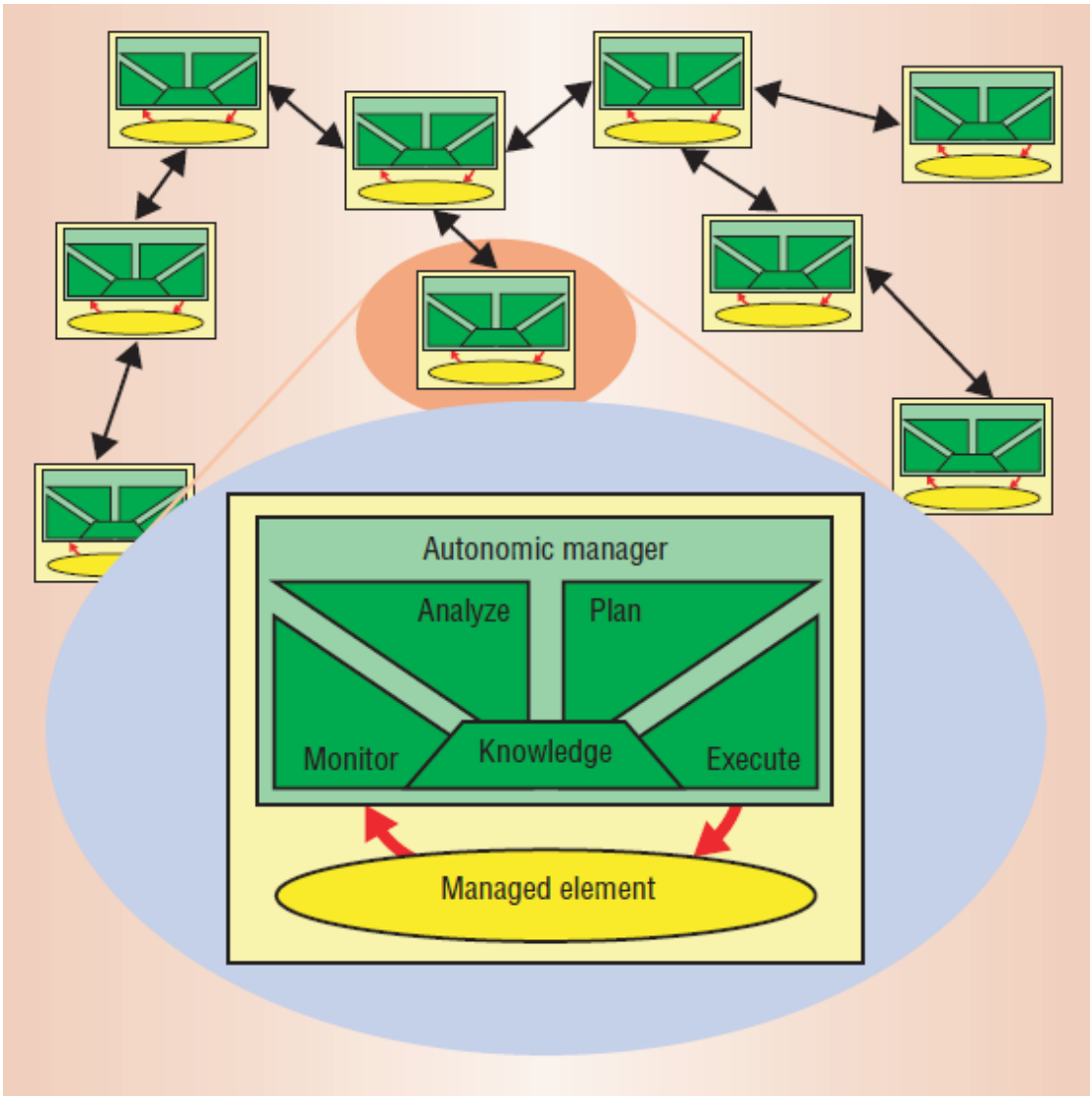

Fig. 2.4 Structure of an autonomic element [Kephart & Chess, 2003]

The ultimate goal of autonomic computing paradigm is to develop computer systems that possess the ability / property / characteristic of self-management system in order to overcome changes (failures, abnormal situation, change of needs, environment changes, etc) during their execution. As described in [Kephart & Chess, 2003], the characteristic include the following:

- Automated configuration of components and systems follow only high-level policies. The rest of the system adjusts automatically and seamlessly.

- Components and system continually seek opportunities to improve their own performance and efficiency.

- System automatically detects, diagnoses, and repairs the localized software and hardware problems.

- System automatically defends against some malicious attacks or cascading failures by using an early warning to anticipate and prevent system-wide failures.

2.3 Self-Organizing Systems

The term self-organizing system refers to a class of systems that are able to change their internal structure and function in response to external circumstances [Banzhaf, 2002]. Its elements are able to organize other elements of the same system by stabilizing the structure or function of the system in response to threats, changes, or fluctuations. Self-organization is an evolutionary process in which the effects of the environment are minimal, where the development of new, complex structures primarily takes place in and throughout the system itself [Anceaume et al., 2005].

[Banzhaf, 2002] provided an overview of self-organizing systems in science, humanities, and engineering by presenting a definition, examples, and roles of self-organizing systems as well as open issues in this research area.

Some characteristics of self-organizing systems are dynamic, open, flexible, adaptive, and resilient. In nature, self-organizing systems have been discovered both in the non-living and the living world, such as galaxies, stars, cells, ecosystems, social systems, immune systems, etc [Banzhaf, 2002], [Hart et al., 2007], in which a global order of the system emerges from local interactions.

According to [Serugendo et al., 2006], there are two trends in building self-organizing systems:

- Biological and nature-inspired algorithms and models are applied for developing self-organizing systems. These models provide a high-level of robustness and adaptation.
- Software engineering approaches are used to define self-organizing and adaptive software architectures in order to provide behavior where components automatically configure their interactions.

Further, they argued that building self-organizing systems on software engineering approaches either as an alternative to bio-inspired techniques, or as a complementary approach may enhance predictability of self-organizing systems and, provide a basis for self-managing systems which must be resilient.

[Krasnogor & Gheorghe, 2005] described and discussed self-assembly systems having similar characteristics to self-organizing systems, especially the inspiration by self-assembly processes and systems in nature to leap forward in technological capabilities, such as fabrication and manufacturing, engineering, computational analysis, and software development.

[Hart, et al., 2007] showed how mechanisms inspired by immunology may offer computational systems that are robust, secure, self-organizing, and self-healing in a manner currently unachievable with the established software engineering techniques.

2.4 Case-Based Reasoning

Case-based reasoning (CBR) means using old experiences to solve new problems [Kolodner, 1992]. It is a process of solving a new problem by remembering a previous similar situation and reusing information and knowledge of that situation [Aamodt & Plaza, 1994]. The foundation of the CBR system is laid on the arguments by [Schank, 1982] on the role of reminding which coordinates past events with current events to enable generalization and prediction. The underlying principle of CBR is that human solve new problems by remembering similar experiences about similar situations.

CBR is able to utilize the specific knowledge of previously experienced, concrete problem situations, called '*cases*'. In it, a new problem is solved by finding a similar past case, and reusing it in a new problem situation. CBR systems store past experiences as individual problem solving episodes as opposed to expert systems, which store past experience as generalized rules and objects [Kolodner, 1992].

CBR also refers to an approach to incremental, sustained learning. Since a new experience is retained each time a problem has been solved, CBR comes to be immediately available for future problems. CBR can either mean adapting old solutions to meet new demands, or using old cases to explain new situations, or using old cases to critique new solutions, or reasoning from precedents to interpret new situation, or creating equitable solution to a new problem [Kolodner, 1992].

Kolodner (1992) listed the advantages of CBR as the following:

- It allows the reasoner to propose solutions to a problem quickly.
- It allows the reasoner to propose solutions in domains that tare not completely understood by the reasoner.
- It gives the reasoner a means for evaluating solutions when no algorithmic method is available for evaluation.
- Cases are useful in interpreting open-ended and ill-defined concepts.
- Remembering previous experience is useful to help learners to avoid repeating past mistakes.
- Cases help the reasoner to focus on its reasoning on important parts of a problem by pointing out what features of a problem are important ones.

Fig. 2.5 shows the CBR cycle as given by [Aamodt & Plaza, 1994]. A new problem is solved by retrieving one or more previously experienced cases, reusing the case in one way or another, revising the solution of a previous case, and retaining the new experience by incorporating it into the knowledge-base (case-base).

A general CBR cycle basically works as follows:

- *Retrieve* the past cases that are similar to the current one.

- *Reuse* the past successful solutions to solve the current problem.
- If necessary, *revise* the proposed solution (adaptation).
- The current experience that is likely to be useful for future problem solving can then be *retained* and stored into the knowledge base (case base).

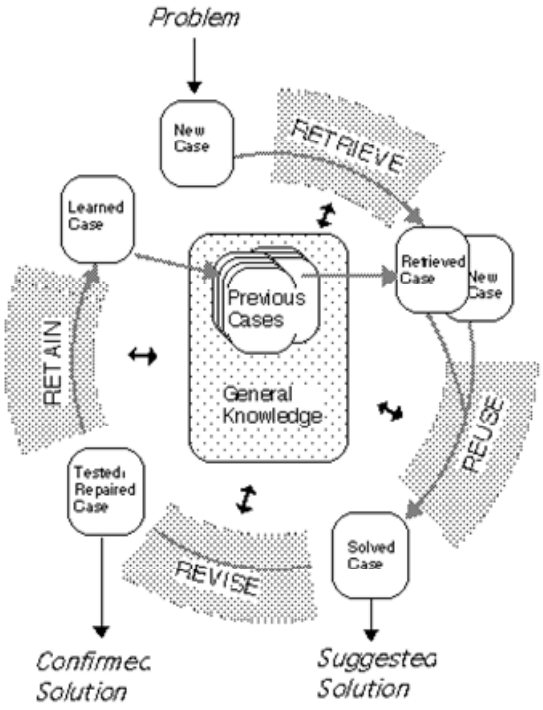

Fig. 2.5 CBR cycle [Aamodt & Plaza, 1994]

Today CBR has been researched in more than 35 institutions all over the world and many applications of CBR have already been put into daily use [Bergman, 2000]. For example, [Cheetham, 2004] and [Morgan et al., 2004] reported deployed CBR applications at GE Plastics and General Motors work places respectively. CBR has also been researched in many different areas such as manufacturing [Hinkle & Toomey, 1995], engineering sales support [Watson & Gardingen, 1999], wireless networks management [Barbera et al. 2002], project management [Xu & Muñoz-Avila, 2004], and fault diagnosis [Yang et al., 2004].

2.5 Related Works

This section presents some works related to this research. To date, several researches have been conducted in the area of autonomic or self-organizing software systems, such as distributed systems, peer-to-peer (P2P) networks, and grid systems; yet so far very few researches has been found to be carried out on autonomic and self-organizing service oriented architecture. Nevertheless, these works would still serve both as foundation and as starting points of this research.

[Georgiadis et al., 2002] examined the feasibility of using architectural constraints as a basis for the specification, design, and implementation of self-organizing architectures for distributed systems. They present a runtime architecture ensuring that after component introduction or failure, the system stabilizes with a structure that satisfies the specified constraints. Some issues however have not been addressed by their work, such as how applications should be designed to take account of the possibility of dynamic service rebinding. This is an important issue in the future SOA since it will need to support dynamic service binding / bonding / composition.

[Montresor et al., 2002] have initiated the Anthill project aimed to design a framework for the development of P2P applications (including grid computing) based on ideas borrowed from complex adaptive systems such as multi-agent systems. The Anthill project uses terminology derived from ant colony metaphor. Related to this research, the Anthill project could contribute an understanding about the adaptation of adaptive systems into distributed systems, such as grid computing and service oriented architecture.

[Champrasert & Suzuki, 2005] proposed an architecture called SymbioticSphere which applies several biological concepts and mechanisms to design grid systems - application services and middleware platforms. The architecture allows data centers to autonomously adapt to dynamic environment changes and survive partial system failures. This work similarly would be helpful for the research in conducting a systematic study of the potential natural / biological process having the desired properties and behavior.

[Zurowska & Deter, 2007] reviewed and presented a model-driven approach of composite web services which is based on the use of Colored Petri Nets. Their work would also be helpful for this research in modeling web services and modeling the interactions with other web services using Petri Nets, and analyzing the models.

[Montani & Anglano, 2008] proposed and described large-scale and distributed software systems with self-healing capabilities using Case-Based Reasoning (CBR). [Gurguis & Zeid, 2005] also proposed a concept to achieve self-healing using web service. As self-healing and self-organizing are parts of the autonomic computing paradigm, these works could also be reference in this research in designing the autonomic and self-organizing service architecture.

The following works attempted to improve the capability and intelligence within SOA domain using software engineering approaches including software agents and semantics. [Maximilien & Singh, 2004] proposed a multi-agents approach that will provide autonomic web services selection which considers the preferences of service consumers, the trustworthiness of providers, semantics, and quality of service. Their approach is based on software engineering architecture and programming model in which agents represent applications and services. [Erl, 2007] introduces service autonomy principle to support the extent to which design principles can be realized in real world environments by fostering design characteristics that increase a service's reliability and behavioral predictability. This principle raises various issues, such as isolation levels and service normalization, which pertain to the design of service logic as well as the service's actual implementation environment. [Schneider et al., 2008] gave an overview of on adaptive service-oriented systems based on software engineering approaches, including service engineering, application engineering, and infrastructure engineering. [Tosi et al., 2009] proposed an approach for designing self-adaptive service oriented applications based on taxonomy of integration faults. Their framework was also inspired by software engineering approach, i.e. computer-aided software engineering.

Other works that are also using agents or semantics include [Ricci et al., 2006], [Ricci & Denti, 2007], [Poggi et al., 2007], [Vitvar et al., 2007], [Shen et al., 2007], and [Balfagih & Hassan, 2010]. Related to this research, those works could provide an insight about the design and development of a more intelligent service oriented framework using software engineering approaches.

2.6 Chapter Summary

It is important to have a solid foundation of the technologies that are utilized and discussed throughout this thesis. The purpose of this chapter is to provide sufficient background information to understand the foundations and concepts of SOC and SOA, autonomic and self-organizing systems, and case-based reasoning technologies that will be elaborated in the rest of this thesis. This chapter starts with introduction on service oriented architecture which includes the topics of current SOA model, web service, and the main roles in SOA: service consumer, service provider, and service registry. It then continues with overview of the enabling techniques for development of adaptive and intelligent computer / software systems, i.e. autonomic computing paradigm, self-organizing systems, and case-based reasoning.

This chapter also discussed some selected related works whose results could be helpful in conducting this research. Those works can provide partial method, technique, or solution to the adaptive and intelligent SOA problem. In this research, their works will be taken into consideration for designing and developing the autonomic SOA framework. This research will be different with those SOA researches in term of the hybrid approach, combining both biological inspired techniques and software engineering approach, as opposed to only employing software engineering approaches in those researches.

CHAPTER 3

METHODOLOGY

3.0 Chapter Overview

This chapter describes the strategy and approach applied in this research, followed by basic structure and architecture of the autonomic SOA, and the techniques and methods adapted in this research.

3.1 Research Workflow

The overall research is divided into several activities, which of each concern with the specific goals to finally fulfill the main objective. The activities list is as follows:

1. Extensive study on service oriented computing, autonomic computing, self organizing systems, case-based reasoning, and processes and mechanisms in biology or nature that present autonomous, self-organizing characteristics. This shall include extensive literature study and thorough analysis of the related works through books, journal and conference papers, websites, etc. The deliverable of this activity is the literature review provided in chapter two.

2. Core concepts of the autonomic and self-organizing theory and mapping autonomic mechanisms into service-oriented computing. The core concepts of autonomic and adaptive systems will be adapted into service-oriented architecture.

It is necessary to identify and define these processes:

- the factors that will enable autonomy of components in SOA
- mechanism to do an optimized search of the required components
- the composition of the service components

3. Architectural framework design and development. An autonomous, self-organizing architectural framework of SOA which is more robust, adaptive, and flexible will be designed and designed. The detail of the proposed autonomic SOA framework is elaborated in chapter four.

4. Evaluation of the concepts and architecture through formal modeling and case studies will be provided. The deliverables of this activity include the Petri Nets and UML models provided in chapter four, and the case studies provided in chapter five.

5. Simulation development and implementation of the prototype in some scenarios of applications as proof of the proposed concept. The deliverables of this activity include the simulation and basic prototype of the proposed autonomic SOA framework. The design and results of the simulation and prototype are provided in chapter five and chapter six.

6. Evaluation and analysis of the simulation and implementation results, the development experience and lessons learned from applying the proposed framework are then analyzed and discussed.

Fig. 3.1 summarizes the workflow of this research.

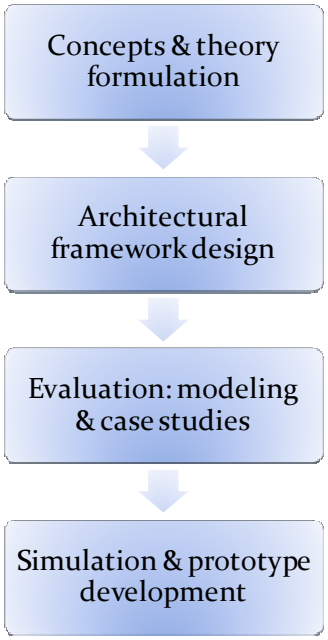

Fig. 3.1 Research workflow

3.2 Case-Based Reasoning Learning Method

Case-based reasoning (CBR) is a process of solving new problem by remembering previous similar situation and reusing information and knowledge of that situation. CBR is able to utilize the specific knowledge of previously experienced, concrete problem situations, called 'cases'. In CBR, a new problem is solved by finding a similar past case, and reusing it in a new problem situation. Since a new experience is retained each time a problem has been solved, CBR comes to be immediately available for future problem solving.

Some benefits of using CBR approach include the following:

- Reasoning by re-using past cases is a powerful and frequently applied way to solve problems (inspired from human's problem solving).
- Being usually easier to learn by retaining a concrete problem solving experience than to generalize from it, CBR favors learning from experience.
- CBR is also known to be well suited for domain where formalized and recognized background knowledge may be unavailable [Montani & Anglano, 2008].

The CBR methodology is chosen in this research because of its benefits over rule-based expert systems which store past experience as generalized rules and objects (as highlighted on the first and second CBR benefit above), and also because in the highly dynamic future SOA, formalized and recognized background knowledge might be unavailable (the third benefit). Case-Based Reasoning (CBR) approach for the learning and adaptation in the framework in this research is adapted. CBR basically works as follows:

- *Retrieve* the past cases that are similar to the current one.
- *Reuse* the past successful solutions and, if necessary, *revise* them (adaptation).
- The current case can then be *retained* and put into the knowledge base (case-base).

A new problem is solved by retrieving one or more previously experienced cases, reusing the case in one way or another, revising the solution based on reusing a previous case, and retaining the new experience by incorporating it into the existing knowledge-base (case-base). Fig. 3.2 illustrates a learning process involving the analysis and planning modules (in autonomic computing paradigm), and knowledge base, adapted from the CBR cycle, i.e. retrieve, reuse, revise, retain.

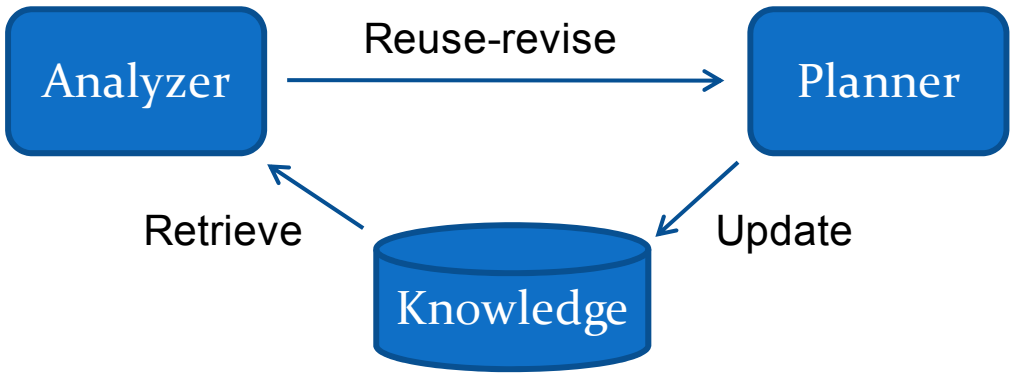

Fig. 3.2 Adaptation / learning using CBR approach

The analyzer will retrieve previous cases whose features include:

- Name and description of service.
- Type of service (atomic or composite).
- If the service is a composite service, then the profile will also include profile of the atomic services needed to compose the composite service (the "ingredients").
- Where, when, how to access (and compose) the service (the "recipe"). The recipe and ingredients are the solution of the case.
- Service usage.

These cases will be analyzed and then reused or revised accordingly to create action plans, and new experience / case will later be updated (retained) to knowledge database.

3.3 Structure of Autonomic SOA

In this research, a structure of the autonomic architecture based on the architectural considerations of autonomic systems given by [Kephart & Chess, 2003] is designed. The autonomic and self-organizing SOA will be a collection of self-organizing elements whose behavior is controlled by autonomic manager. Each self-organizing element will manage its behavior, internal services, and relationships with other self-organizing elements and also can request services from and provide services to the other self-organizing elements. The self-organizing SOA managers will do the four main processes: monitor, analyze, plan, and execute (i.e. autonomic cycle).

The managers will do those processes using the knowledge database, interacting with both their managed services and the external environment (other self-organizing SOA managers) to provide their services or come up with new services (emergence). Fig. 3.3 shows the basic structure of the proposed autonomous SOA. The details of the proposed framework will be elaborated later in chapter four.

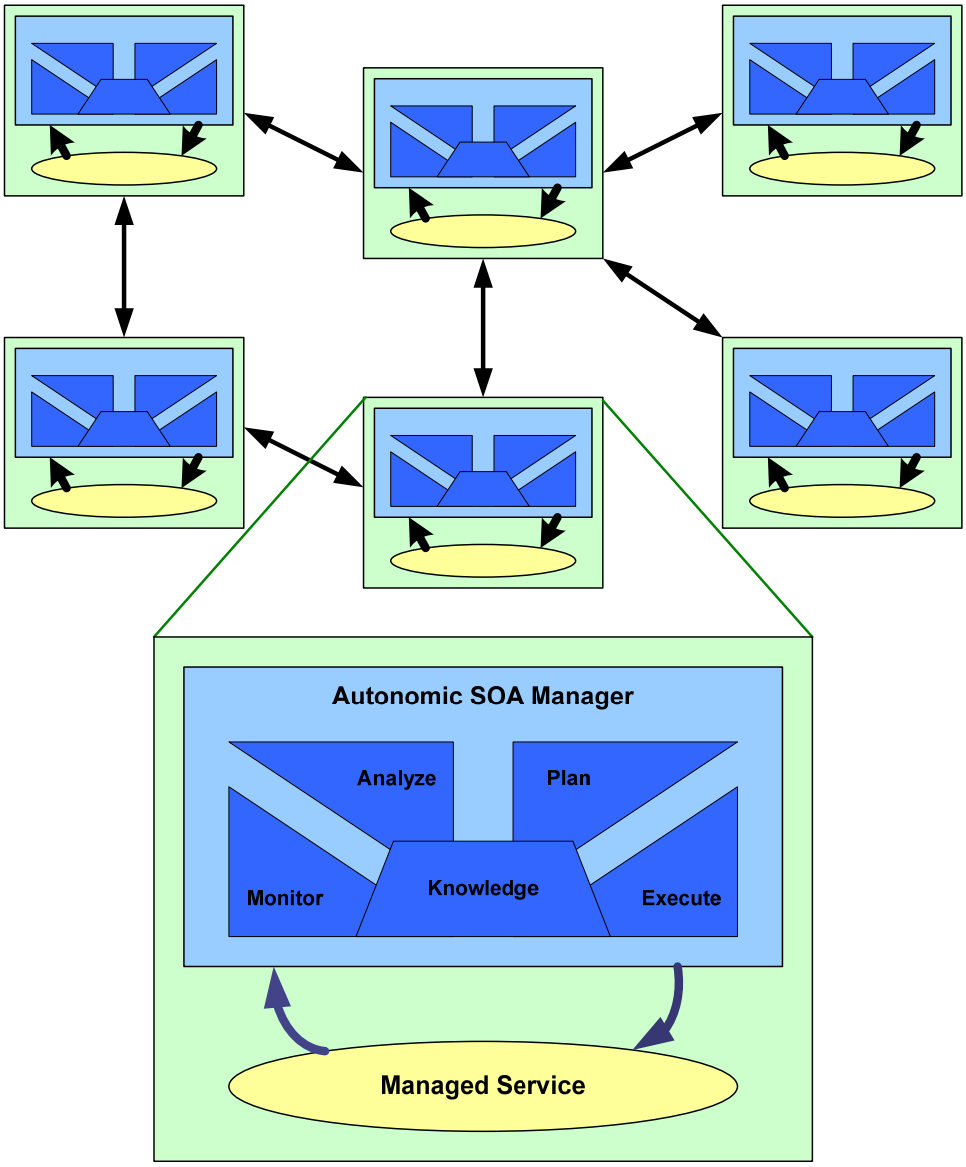

Fig. 3.3 Structure of autonomous SOA (adapted from [Kephart & Chess, 2003])

3.4 Metrics

[Russell et al., 2006] described the following attributes - measurements that can be applied to the system to determine its overall dependability.

- Availability – the probability that a service is present and ready for use:

$$Availability = \frac{uptime}{uptime + downtime}$$

- Reliability – the capability of maintaining the service and service quality:

  $Reliability = e^{-\lambda t}$

  where λ = *failure rate*

- Performance – defined in terms of throughput of the services and latency.
- Maintainability – to undergo modifications, repairs, and can evolve over time.
- Safety – the absence of catastrophic consequences.
- Confidentiality – information that is accessible only to those authorized to use it.
- Integrity – the absence of improper system alterations.

Those attributes can be used to test and measure the performance of the proposed architecture. Note that some attributes are quantifiable by direct measurements while others are more subjective. For instance Safety cannot be measured directly via metrics but it is a subjective assessment that requires judgmental information to be applied to give a level of confidence, whereas Reliability can be quantified by physical measurements.

In this field, the quantifiable aspects of dependability, for instance Availability and Reliability, are the only ones to focus on. In this regard, the last three attributes (i.e. Safety, Confidentiality, and Integrity) are those related to security issues that are beyond the scope of the research. Hence the tests and measurements of those attributes will not be considered.

Other metrics that can be used to test an architecture include the following:

- Accuracy: how accurate the system provides the requested services (compared to user's service requirements / descriptions).
- Response time: how long it takes for the system to provide the requested services.
- Agility: one measure is to compare the time it took to complete applications in the past with the time to completion under SOA (in this case: compare the

time it took to complete applications in the typical SOA with the time to completion under SOA with self-organizing feature).

- Service consumption, such as:
  - The number of service consumers
  - Service usage by consumers
  - The minimum, average, and maximum response times
- Usability as one attribute of system design, based on the works in [Bass et al, 2001], [Goudar, 2008]. The metric measures the usability of the services by various consumers across multiple channels. These measures give an indication on how the service infrastructure is being used by various consumers. In cases where service consumptions are being billed as per the consumption, these measures form basis to calculate the service billing

3.5 Affinity Characteristics

Adapted from the object oriented paradigm with some additions and removals, the characteristics of relationship (affinity) between elements in service oriented paradigm are described in this section and will be used later to determine the metrics to measure the affinity of the elements in SOA. In contrast to the object oriented paradigm, service-oriented paradigm introduces an additional level of abstraction and encapsulation: a service, in which operations (methods) are aggregated into elements (classes, business process scripts, procedural packages, etc.) that implement the functionality of the service as exposed via operations in the service interface. On the other hand, many categories in object oriented paradigm should be omitted due to their incompatibility with service-oriented paradigm [Perepletchikov et al., 2007].

The affinity characteristics are the following:

- Common Data (C_DATA): service interface operations using the same input parameters.
- Common Usage (C_USAGE): service interface operations being used by the same consumers. A typical service consumer would be a business process, running either within or outside the system boundaries.

- Common Sequence (C_SEQ): service interface operations being invoked sequentially from service consumers, where a post condition/output of a given operation satisfies a precondition/input of the next operation.
- Common Implementation (C_IMPL): service interface operations being implemented by the same implementation elements.

Some affinity metrics, also proposed in this research, to measure the affinity characteristic mentioned previously, are described below:

- Service Interface Data Affinity (SIDA)
  SIDA metric quantifies affinity of a given service based on the affinity of the operations exposed in its interface, as reflected by these operations sharing the same parameter types. A service is deemed to be highly related when all service operations work on the same input parameter types.
  SIDA is based on C_DATA characteristic.

- Service Interface Usage Affinity (SIUA)
  SIUA metric quantifies affinity of a given service based on the affinity of the operations exposed in its interface, as reflected by the behavioral communication (usage) pattern of service consumers. A service is deemed to be highly related when all service operations are invoked by every client (service consumer).
  SIUA is based on C_USAGE characteristic.

- Service Sequential Usage Affinity (SSUA)
  Similar to SIUA, SSUA metric quantifies affinity of a given service based on the affinity of the operations exposed in its interface, as reflected by the behavioral communication (usage) pattern of service consumers. The difference is that in the case of SSUA the dependencies among service operations are taken into consideration. More specifically, the communication is deemed to be sequential if the output from one operation serves as the input for the next operation or the post condition of an operation satisfies the

precondition of the next operation. SSUA is based on C_USAGE and C_SEQ characteristics.

- Service Implementation Affinity (SIA)
  This quantifies affinity of a given service based on the cohesiveness of the operations exposed in its interface, as reflected by the associated implementation elements. A service is deemed to be highly related when all service operations are implemented by the same implementation elements. SIA is based on C_IMPL characteristic.

- Total Service Affinity (TSA)
  TSA refers to a combination of all possible values of the mentioned affinity (SIDA, SIUA, SSUA, and SIA) as reflected by operations exposed in service interface.

3.6 Petri Nets based Validation Methodology

Petri Nets [Petri, 1966], [Murata, 1989] - based functional validation framework is used to analyze the SOA framework proposed in this research. This framework was introduced by [Yoo et al, 2009] to validate service composition in SOA. Fig. 3.4 shows the functional validation methodology. Later on, the state transitions using Petri Nets modeling will be analyzed for enabling the process of validation on service's behavioral correctness and other properties.

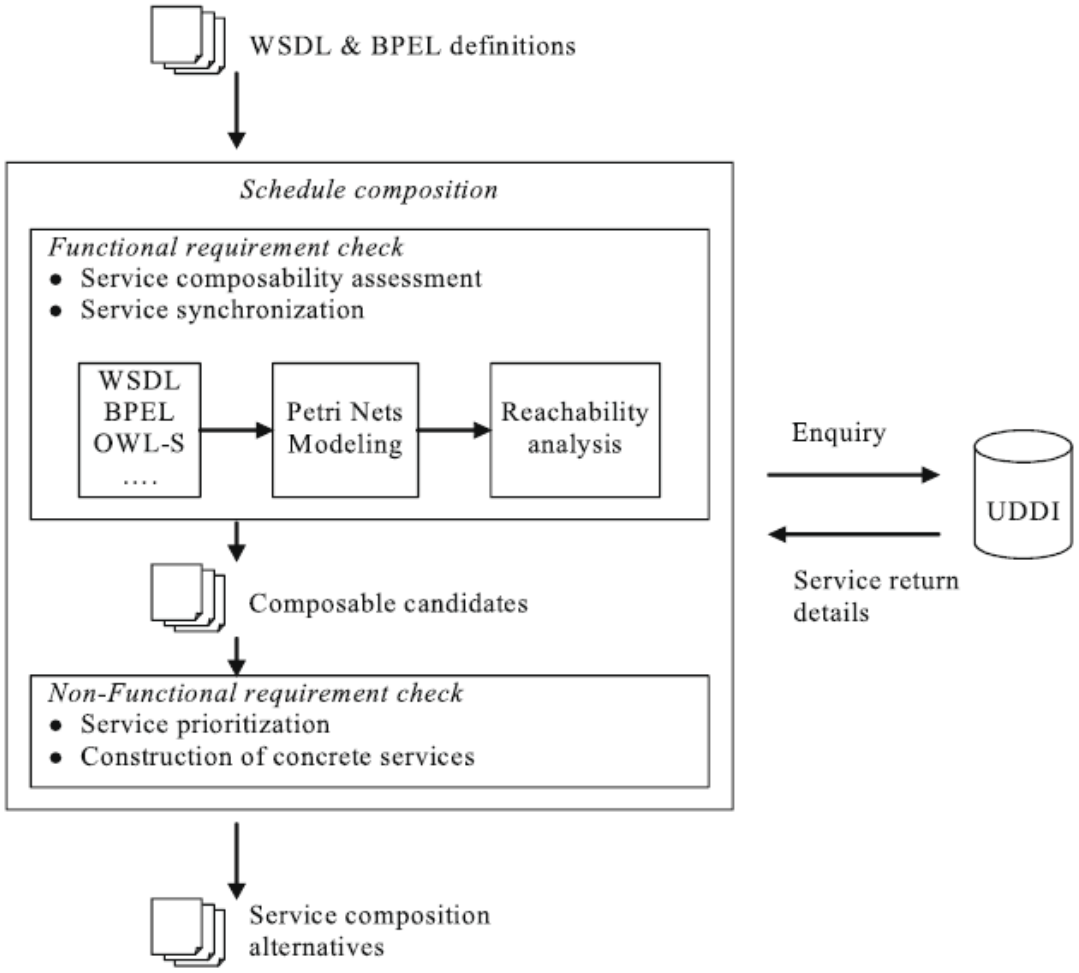

Fig. 3.4 Functional validation methodology [Yoo et al, 2009]

3.6.1 Coloured Petri Nets

Coloured Petri Nets (CPNs) [Jensen, 1997] is a modelling language developed for systems in which communication, synchronization and resource sharing play an important role. CPNs combine the strengths of ordinary Petri nets with the strengths of a high-level programming language. Petri nets provide the primitives for a process interaction, while the programming language provides the primitives for the definition of data types and the manipulations of data values.

CPN models can be made with or without explicit reference to time:

- Untimed CPN models are usually used to validate the functional/logical correctness of a system.
- Timed CPN models are used to evaluate the performance of the system.

CPNs also offer more formal verification methods, i.e. state space analysis (reachability, boundedness, home properties, liveness, and fairness) and invariant analysis. A transformation technique proposed by [Zhanhg & Zhu, 2009] is also used in this research to derive the CPN model from UML diagram.

3.6.2 CPN Tools

CPN Tools is a tool for editing, simulating and analyzing Coloured Petri Nets (CPN) developed by CPN Group, University of Aarhus, Denmark. The Graphical User Interface is based on advanced interaction techniques, such as tool-glasses, marking menus, and bi-manual interaction. Feedback facilities provide contextual error messages and indicate dependency relationships between net elements.

The tool features incremental syntax checking and code generation which will occur while a net is being constructed. A fast simulator efficiently handles both untimed and timed nets. Full and partial state spaces can be generated and analyzed, and a standard state space report contains information such as boundedness properties and liveness properties. The functionality of the simulation engine and state space facilities are similar to the corresponding components in Design/CPN, which is a widespread tool for Coloured Petri Nets.

The purposes of using CPN tools in this research are:

- To provide a description / specification of the autonomic SOA.
- To analyze the behaviour of the developed CPN model of the autonomic SOA:
    - Modelling using CPN tools is interactive and automatic
    - Formal analysis method
- To provide an improved understanding of the autonomic SOA.

3.7 Chapter Summary

This chapter has presented the research workflow, methodology, and approach employed in this research. It additionally presents the structure and architecture of the proposed autonomic, self-organizing SOA. Some processes in nature and biology having interesting properties to be adapted into the autonomous SOA are studied as well. A decision to adapting the autonomic computing paradigm and case-based reasoning eventually was drawn. This decision was based on the facts that autonomic computing paradigm and case-based reasoning, as inspired by biological systems (i.e. human nervous system and human problem solving) provably could provide autonomic and self-management capabilities to computer systems. And it is also because to invent a completely new nature-inspired model would require in-depth knowledge in that nature process including the biological and chemical processes behind them, which will be out of the scope of this research.

Identifying the affinity characteristics of the service oriented paradigm, especially in contrast with the object oriented paradigm, before proposing the affinity metrics based on those characteristics, was also conducted in this research. The metrics in turn can be used to measure the affinity of the elements in SOA. Other quantifiable metrics to measure SOA performance have also been identified. Further, identifying the methodology that is going to be used to analyze the proposed framework, i.e. Petri Nets-based validation methodology, was the last part of this chapter.

CHAPTER 4

ARCHITECTURAL FRAMEWORK

4.0 Chapter Overview

In this chapter, to show and describe the proposed architecture and formally specified the architecture, models of the proposed architecture using Unified Modeling Language (UML) and Petri Nets based modeling methods were elaborated.

4.1 Autonomic Service Oriented Architecture

Based on the aforementioned SOA study in the previous chapters, a initial design of the architecture of proposed system in this research as illustrated in Fig. 4.1 is presented. The architecture consists of three main entities, including

- Service requestor
- Service provider
- Service registry and broker consisting of service registry, service broker, and service aggregator systems.

It is also possible that service requestor will also act as service provider and vice versa. The autonomic, self-organizing SOA will provide service requestors with atomic service, i.e. self-contained service and do not invoke any other services, or composite service, i.e. a service whose implementation calls other services composed from various atomic web services.

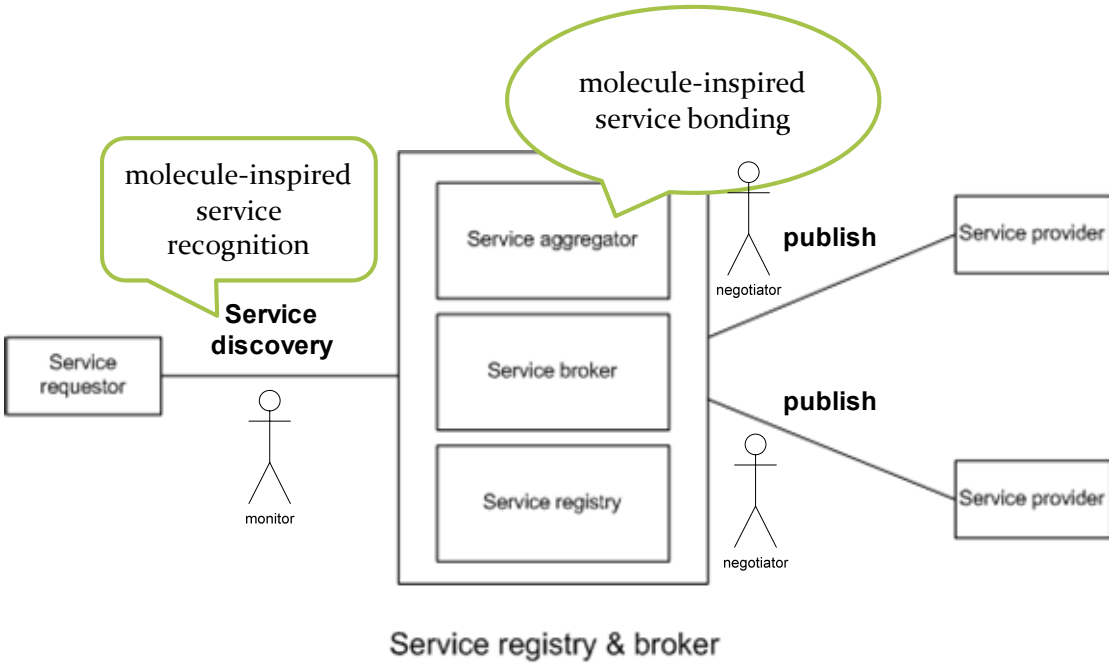

Fig. 4.1 Initial service oriented architecture design

The initial architecture was designed based only on SOA paradigm. It has the components of typical SOA framework; however it still does not have the features of autonomic computing paradigm. Therefore the initial architecture was then extended (using layered approach) to incorporate autonomic computing paradigm. Fig. 4.2 illustrates the overall extended architecture of the proposed autonomic SOA system.

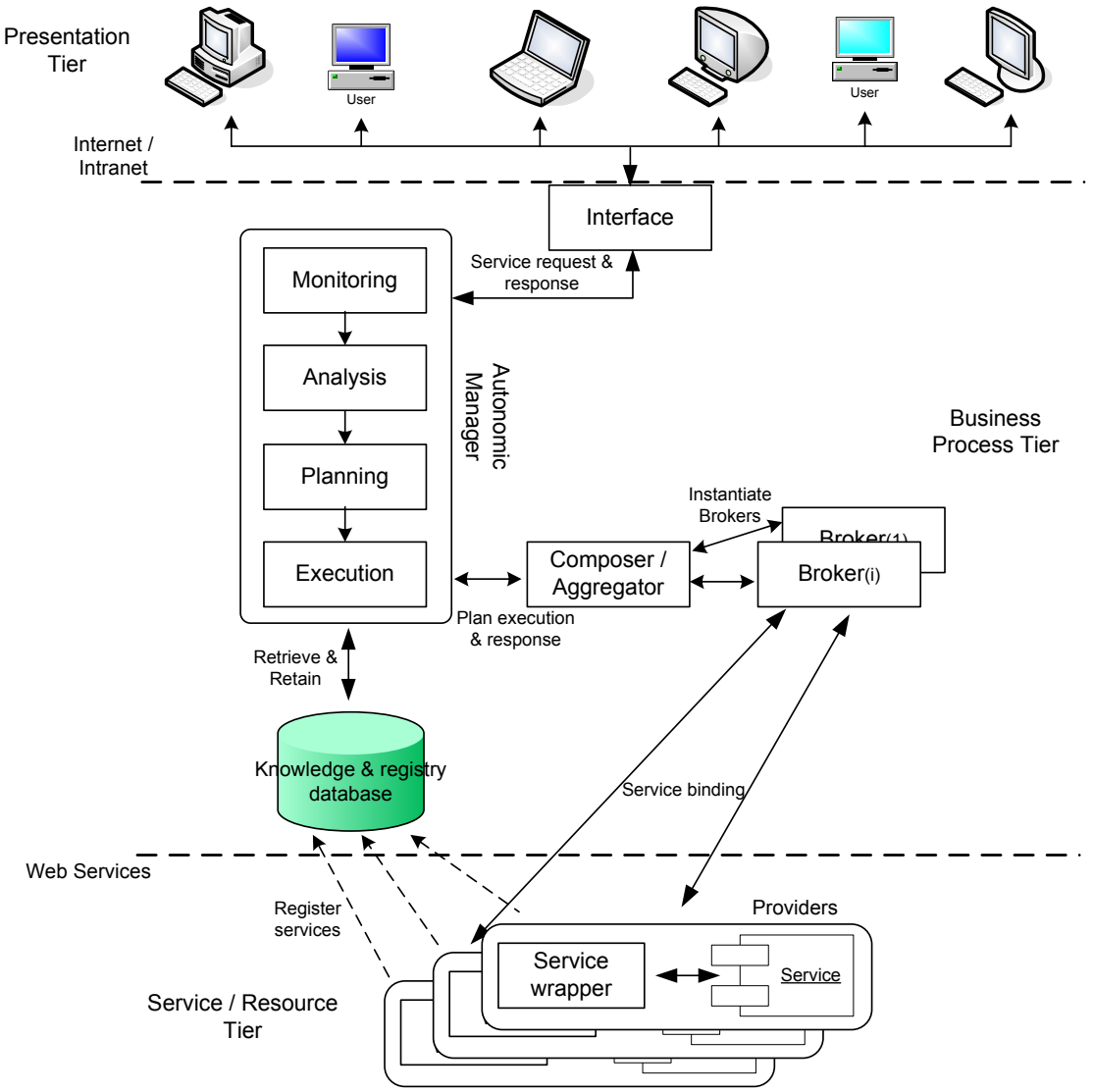

Fig. 4.2 Overall architecture of the autonomic SOA

The architecture is separated into the three tiers:

- The top that is a presentation tier to provide various users through web
- The mid that is a processing tier to perform and coordinate several jobs and
- The bottom that is a service / resource tier to enable the utilization of the distributed resources via Web Services.

The service/resource tier refers to service providers in a typical SOA framework. The brokers in processing tier act as service requestors. Here, the functionality of the service registry, by adding a knowledge base as required by the autonomic computing paradigm, is extended. The knowledge base provides the capability to store the previous services profiles (cases) whose features include:

- Name of the service.
- Description of the service.
- The type of service (atomic, composite).
- If the service is a composite service, then the profile will also include profile of the atomic services required to compose the composite service ("ingredients").
- Where, when, how (sequence) to access (and compose if necessary) the service ("recipe").

The autonomic computing paradigm is incorporated in the processing tier which has the autonomic manager in it. In the context of autonomic computing paradigm, the autonomic manager will perform the autonomic cycle, i.e. monitoring, analyzing, planning, and executing, which of each is described in the following sections.

4.1.1 Monitoring

The manager will monitor both its own behavior and the overall system, including the following:

- The availability of the services,
- Addition of new services,
- Removal of services,
- Request / query from user, etc.

A sentinel or monitoring module will provide monitoring services to the elements. Along with service registry, it would provide service discovery. The service monitor

continuously monitors the system to detecting and identifying request from user and the status of services. If a service request input is available from user, it will be forwarded into analysis. Then if there is a change in service status, the status of that particular service in knowledge base will be updated. A change in service status will be considered as a new request that will be treated as such (forwarded to analysis module and so forth). The monitoring process is illustrated in Fig. 4.3.

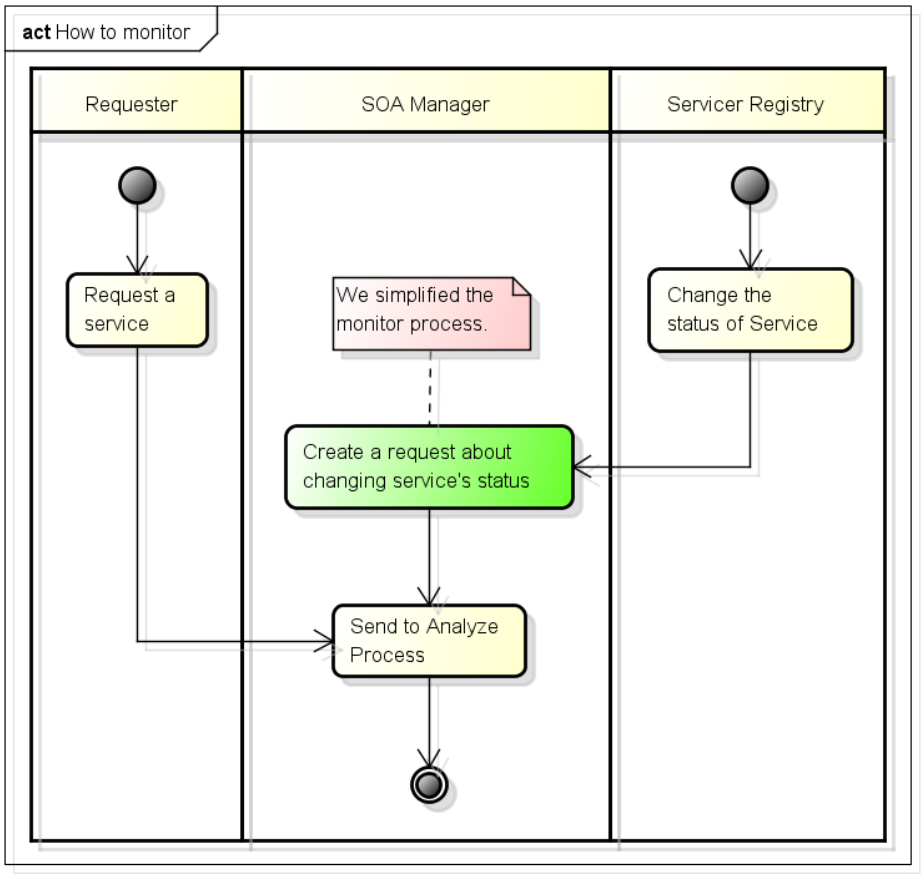

Fig. 4.3 Monitoring process

4.1.2 Analyzing

It means to analyze the requests. The manager will retrieve previous cases from the knowledge base, whose features include description of services, type of service (atomic or composite), their providers, and access to the providers. The cases then will be reused - revised as necessary to provide the (composite) service requested.

Fig. 4.4 illustrates the analysis process adapted from the CBR (Case-Based Reasoning) cycle (retrieve, reuse, revise, and retain) for adaptive and learning functionality, which include both the analysis and planning processes using the knowledge base as the case base. For its benefits, features, and successful implementation in the autonomic system found in [Montani & Anglano, 2008], CBR here becomes the chosen in the analysis and planning processes. Fig. 4.5 illustrates the adaptation of CBR and autonomic computing cycle in the proposed autonomic SOA framework.

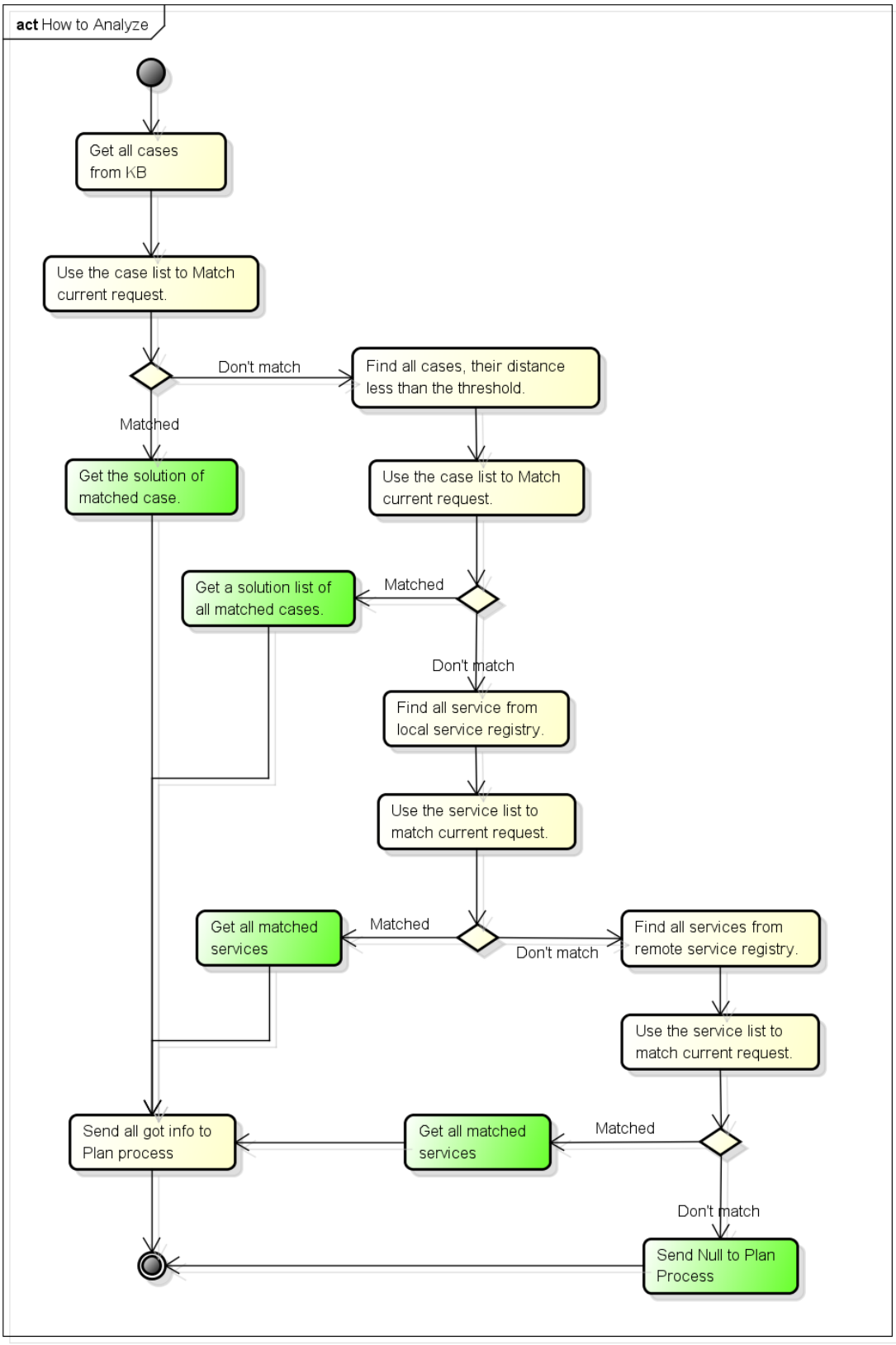

Fig. 4.4 Analysis process

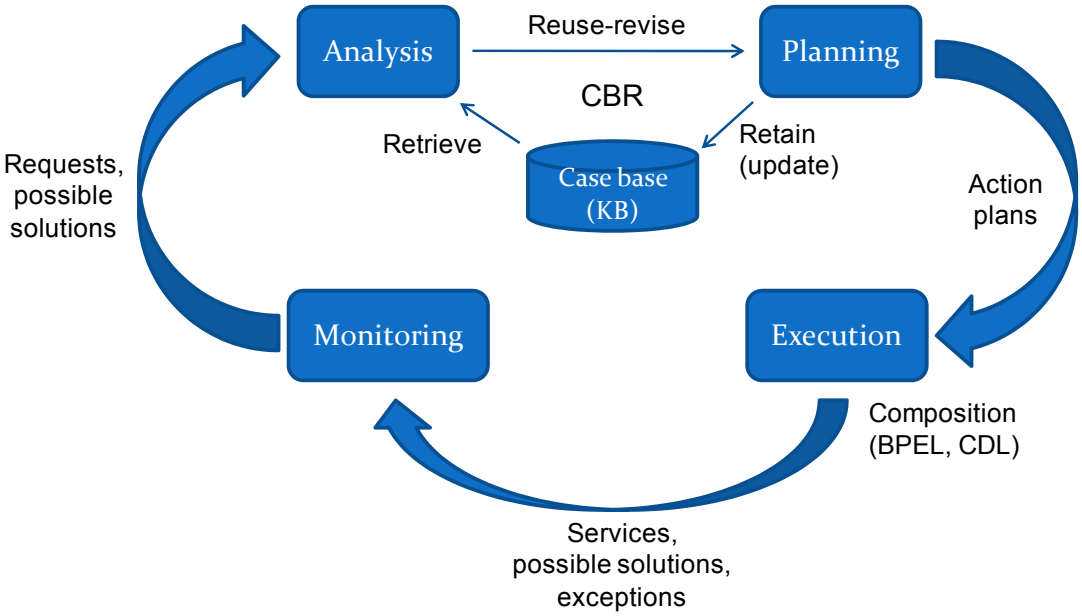

Fig. 4.5 Adaptation of autonomic cycle and CBR in autonomic SOA

The analysis process is described as the following:

- Once receiving a request of service, the system starts by first searching for that particular service profile (as represented by a case) in knowledge base / case base. If that particular service profile is available, then it is retrieved for action planning.

- If there is no service profile of that particular service in the knowledge base, then cases that are having similar properties / features would be retrieved. Various metrics can be used to calculate the similarity distance. For example, the work by [Montani & Anglano, 2008] used heterogeneous Euclidian-overlap metric (HEOM) [Wilson & Martinez, 1997] as the following:

  If *f* is a feature, then:

$$HEOM = \sqrt{\sum_{f} d_f(x, y)^2}$$

Where:

$d_f(x, y) = 1$, if x or y are missing

$d_f(x, y) = overlap(x, y)$ if f is symbolic feature, (i.e. 0 if $x = y$, 1 otherwise)

$d_f(x, y) = \frac{|x-y|}{range_f}$ if f is a linear feature

$range_f = max_f - min_f$

The distance calculation returns a value which is typically in the range of 0..1 with 0 value means zero distance, i.e. $x = y$.

- The similar cases found shall be used for action planning (by revising them). The new case afterward will be used for action planning and then added to the knowledge base.

- If there are no similar previous cases, the monitoring module will search for the composite service in service registry (or search for atomic services that could be composed into the requested service). For scalability, the system should also be able to search in other service registries (e.g. online service registry on the internet or other service ecosystems) if the local service registry does not have the services needed. The new service profile will then be used for action planning and added (retained) to the knowledge base.

- The autonomic manager will also suggest other services to the users who are related to the requested services (e.g. other services that are also typically used) based on the previous cases in the knowledge base.

The mechanisms of the CBR in the autonomic SOA are described in the following algorithm:

- **Overall CBR mechanism:**

  Algorithm overview:

  The system will retrieve every record from knowledge base (KB) by firstly trying to find exact match of the current case in those records. If an exact match is found, the solution then is forwarded to the next phase, yet if not, the system will select cases that are similar with the current case. The solutions of those selected cases (list of possible solutions) are forwarded to the next phase. However, if there are no similar cases found, the system will search for the service at external / remote service registries.

  Input: N number of cases, case(current)
  Output: solution(current), listOfSolutions
  Internal: listOfCases
  start
  listOfCases := retrieve(N)
  solution(current) := reuse(listOfCases, case(current))
  if solution(current) ≠ {} then
  compose solution(current)
  else
  listOfSolutions := revise(listOfCases, case(current))
  if listOfSolutions ≠ {} then
  compose listOfSolutions
  retain(case(current))
  end if
  end else
  end

- **Retrieve mechanism:**

  Algorithm overview:

  Retrieving every record in KB (and put them in an array / list).

  Input: N number of cases in KB
  Output: listOfCases

```
start
for ∀i ∈ N
        read case(i)
        listOfCases = listOfCases + case(i)
end for
return listOfCases
end
```

- **Reuse mechanism:**

  Algorithm overview:

  Find an exact match by comparing every record with the current case (or find the case with zero distances to the current case). If it is found, then return that record's solution as the current solution.

```
Input: N number of cases in KB, listOfCases, case(current)
Output: solution(current), initially empty
Start
solution(current) := {}
for ∀i ∈ N
        if case(i) == case(current)
                solution(current) = solution(i)
                break loop
        end if
end for
return solution(current)
end
```

- **Revise mechanism:**

  Algorithm overview:

  Calculate the distance (d) between every record and current case. If the distance is 0, it means that it is an exact match, and then return that record's solution as the current solution. If there is no case with 0 distances, select

cases with distances below the distance threshold and save their solutions as a list of possible solution, and forward it to the next phase.

Input: N number of cases in KB, listOfCases, case(current)
Output: listOfSolutions, initially empty
Start
listOfSolutions := {}
for $\forall i \in N$
    if d(case(i), case(current)) < threshold then
        listOfSolutions = listOfSolutions + solution(i)
end

Eventually, solution / recipe that is accepted by users (i.e. used by many users, high usage numbers) will be retained, while other solutions with low usage numbers will be discarded from KB.

- **Retain mechanism:**
  Algorithm overview:
  Record new or updated cases and service status.

  Input: case(current)
  Output: stateUpdate
  start
  record case(current)
  record stateUpdate
  send stateUpdate to other element
  end

At the end of the retain mechanism, there will be a *stateUpdate* process if there are new cases to be retained. This process is required by the snapshot mechanism that will be discussed in section 4.4.

4.1.3 Planning

Autonomic manager will plan actions to provide the requested composite service. It plans the suitable actions for the requested service. If it is a composite service, then the action plans will include the following:

- The list of available atomic services needed to compose the required composite service
- Where and how to access the atomic service
- The sequence of accessing the atomic service

It will also update the knowledge base if new action plan is created (or revised from the previous ones) so that these plans can be readily available and prepared faster when the same composite service is re-requested in the future. The planning process is illustrated in Fig. 4.6. After receiving the service information from analysis module, the planning module will either create an action plan to invoke the service solution or it will create several action plans of the previous similar cases. The action plan(s) will then be forwarded to execution module.

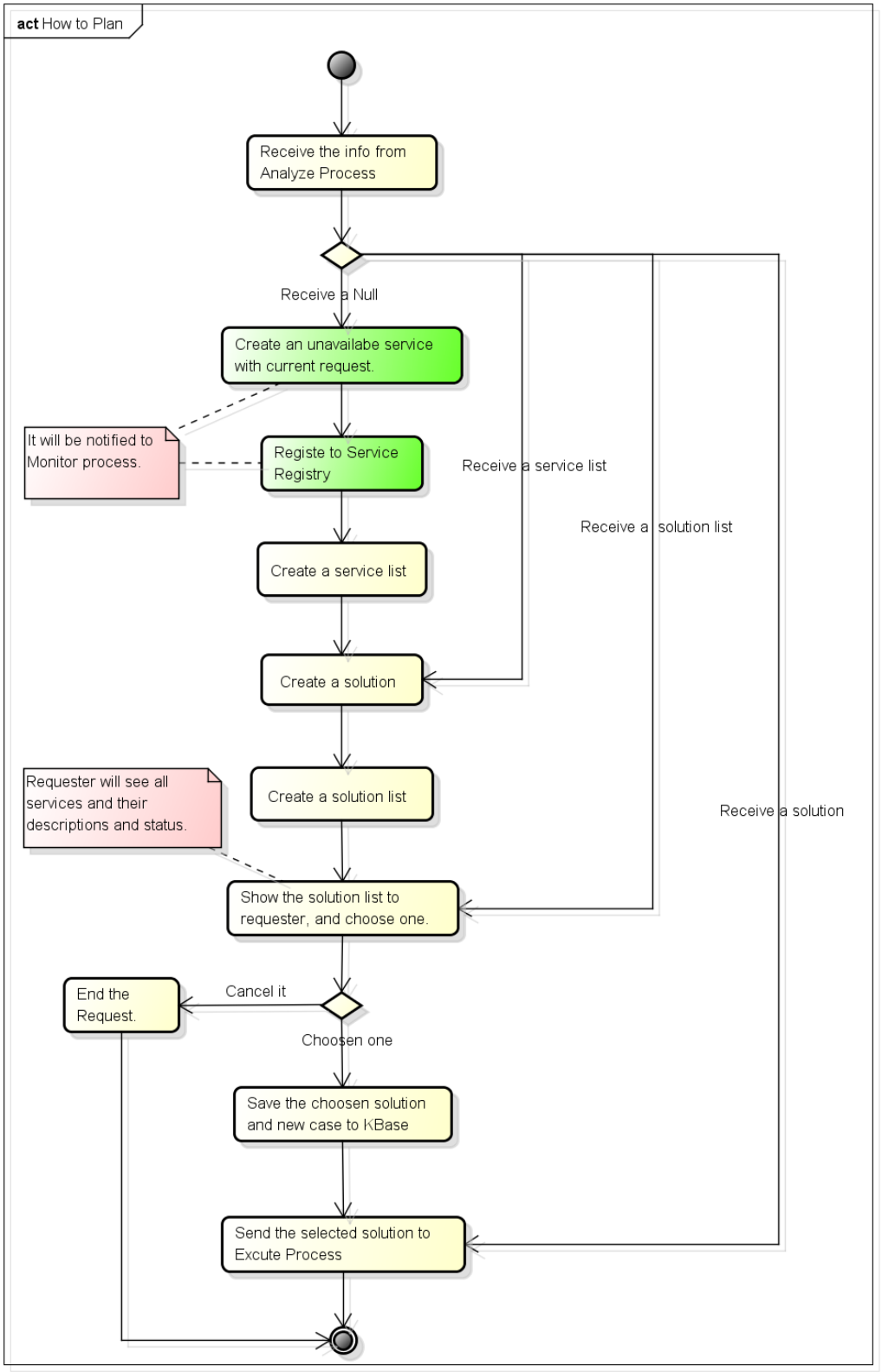

Fig. 4.6 Planning process

4.1.4 Executing

Autonomic manager will execute a plan to provide a requested service, and brokers will assist in interacting and negotiating with the service providers to obtain the required services, including translating messages from the formal messaging protocol of the sender to the formal messaging protocol of the receiver if necessary (in the case where sender and receiver are using different platforms). Upon receiving action plan, the execution module will execute it utilizing the brokers as necessary to interact with service providers.

If the requested service is an atomic service, then the service will be simply provided by the service provider. Meanwhile if it is a composite one, then the autonomic manager will execute the action plan and then provide the composite service, which is by composing the atomic services that can be based on Business Process Execution Language (BPEL) or Choreography Description Language (CDL). The execution process is illustrated in Fig. 4.7.

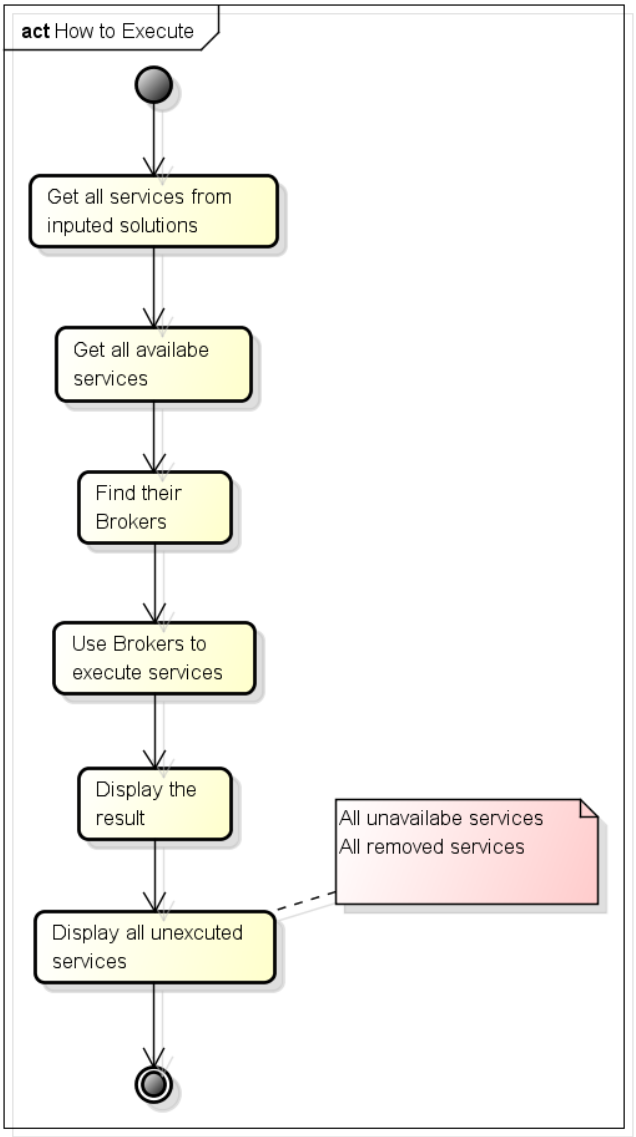

Fig. 4.7 Execution process

4.2 Meta-Modeling

In this research the Unified Modeling Language (UML)-based SOA meta-modeling concept [Zhang et al., 2006] to modelling the proposed architecture in UML class diagram is used. This approach enables meta-modeling of SOA using the notations of UML. On the basis of generic component-based modeling techniques, the notion of service component is adapted for describing the fundamental building block in SOA and its meta-model is identified as well.

The meta-model of the proposed architecture is illustrated in Fig. 4.8. It is noted that the class diagram shows two instances of *Broker* and three instances of *Provider* as the illustrative examples. In actual implementation, the number of instances of *Broker* and *Provider* could be more.

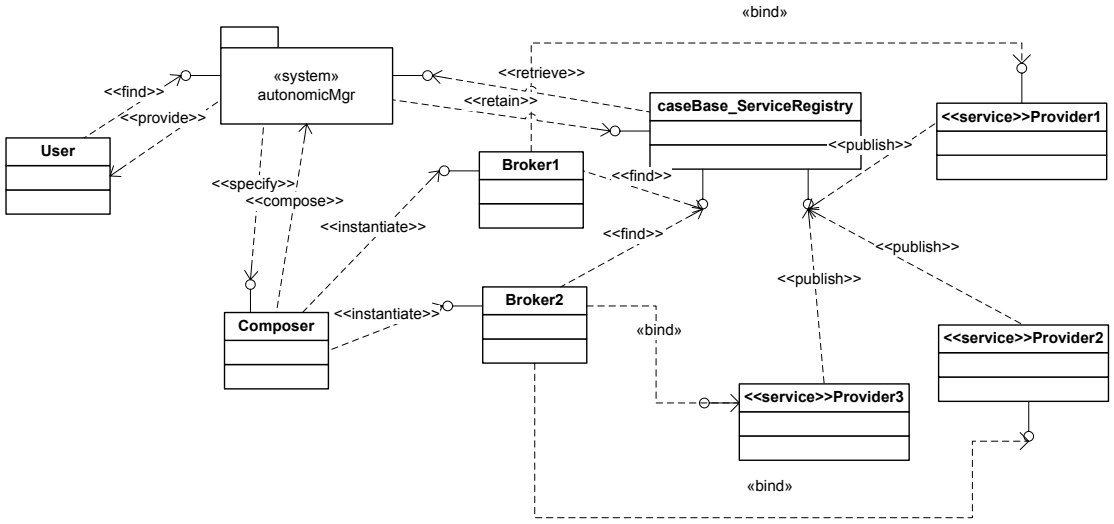

Fig. 4.8 Meta model of the autonomous SOA in UML class diagram

To specify the relationships between the components, the following different stereotypes adapted from [Zhang et al., 2006] and the Meta-Model for SOA [Everware-CBDI, 2011] are used:

- Stereotype <<*request*>> and <<*provide*>> to specify the relationship between user and the autonomic system
- Stereotype <<*call*>> to specify the relationship between the autonomic, self-organizing manager and composer
- Stereotype <<*instantiate*>> to express the relationship between composer and its brokers
- Stereotype <<*publish*>> to express the relationship between service providers and service registry
- Stereotype <<*find*>> to express the relationship between brokers and service registry

- Stereotype <<*bind*>> to express the relationship between brokers and service providers

Having specified the relationship between the components above, the UML sequence diagram of the proposed architecture as illustrated in Fig. 4.9 is derived.

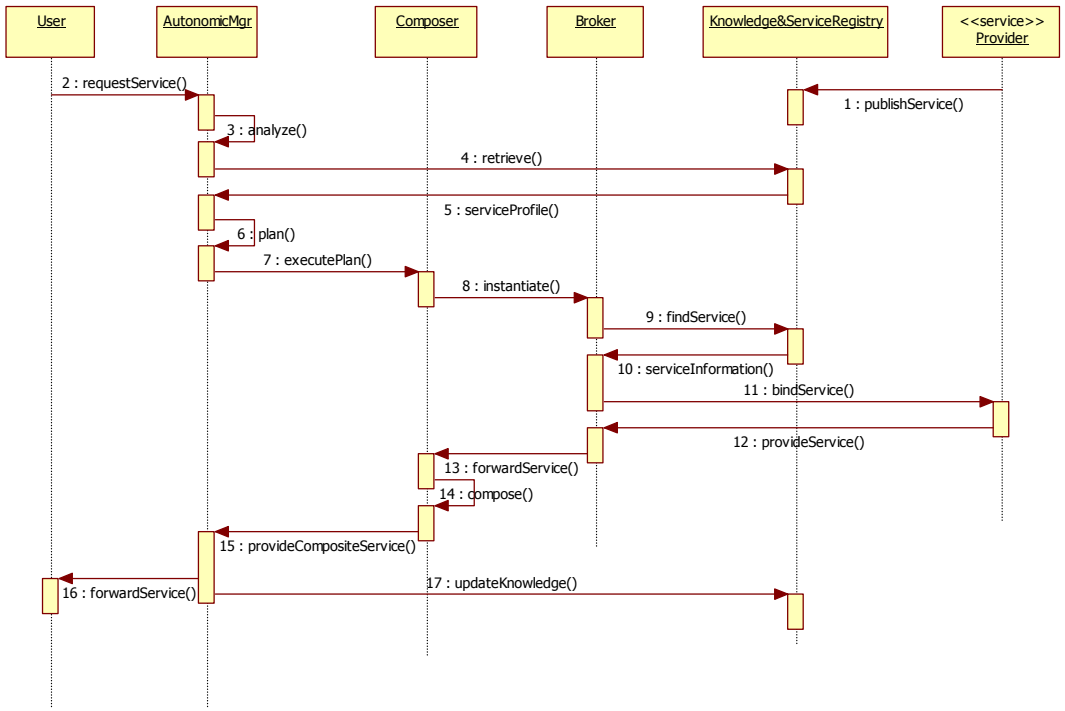

Fig. 4.9 UML sequence diagram of the autonomic SOA

Note that in the sequence diagram above, the monitoring process is running continuously and is not shown. In the case of a new profile of composite service, the sequence in addition might loop back to the analysis process if the new composite service does not match with the user's criteria / description / requirements.

4.3 Formal Definitions

A SOA (Service-Oriented Architecture) is "*a set of components which can be invoked, and whose interface descriptions can be published and discovered*" [Booth et al., 2004]. The services are published by service providers. Service requestors then can discover (via service registry) and invoke those services. Thus SOA could be formally defined as follows:

Definition 1: *SOA* is a four-tuple, consisting of service (*S*), service provider (*SP*), service requestor / consumer (*SC*) and service registry (*SR*):
$SOA = \langle S, SP, SC, SR \rangle$

Service (*S*) refers to an abstract resource that represents a capability of performing tasks [W3C, 2004]. It additionally can be considered as a container for a set of system functions that have been exposed to the web-based protocols. There are two types of services [Woolf, 2006]:

- Composite service (S_c) - a service whose implementation calls other services and as a result of composition function, *c*, result of other services
- Atomic service, (S_a) - self-contained service and not invoking any other services

Definition 2: $S_c = c(s_1, \ldots, s_n)$, where $\{s_a, s_c, s_1, \ldots, s_n\} \in S$

Service registry is an authoritative, centrally controlled store of service information [W3C, 2004], which may be used by both service providers to publish their services and service requestors to discover services using Web Service Description Language (WSDL). Thus:

Definition 3: $SR = \{desc(s_1), \ldots, desc(s_n)\}$
where $desc(s)$, is a description of service, $s \in S$

Service provider (*SP*) is the entity that provides service [W3C, 2004]. Thus it can be considered as a set of service.

Definition 4: $SP = \{s\}, s \in S$

The autonomic manager will carry out the autonomic computing cycle: monitor, analyze, plan, and execute, via its knowledge base [Kephart & Chess, 2003]. Hence:

Definition 5: Let autonomic manager, *AM,* be a 5-tuple, consisting of monitoring (*M*), analysis (*A*), planning (*P*), execution (*E*) and knowledge base (*KB*):

$AM = \langle M, A, P, E, KB \rangle$

Autonomic systems consist of autonomic elements, whose behavior is controlled by autonomic manager [Kephart & Chess, 2003]. Therefore the Autonomic SOA, (*ASOA*) is an SOA with autonomic manager (*AM*):

Definition 6: $ASOA = \langle SOA, AM \rangle$

Using definition 1, 5, and 6, it is obtained:

Definition 7: $ASOA = \langle SP, S, SR, SC, M, A, P, E, KB \rangle$

In the proposed framework, *AM* will retrieve *cases* from *KB*, whose features include:

- Name of the service
- Type of service (atomic, composite, etc)
- Description of service (WSDL-based)
- Number of usage (to measure the usability of the service)
- The "recipe" as the solution:
    - The list of service providers or "ingredients" of a service
    - How to access the service providers and to compose a composite service

Definition 8:

case = ⟨name, type, desc, usage, solution⟩

4.4 Snapshot Mechanism

To achieve more robust service oriented architecture and to reduce service searching time, a mechanism to determine global status of services in the autonomic elements is necessary. To determine a global status, an autonomic element (*ae*) in turn must enlist cooperation of other elements that must record their own local services status and send the recorded local status to *ae*.

The work in [Chandy & Lamport, 1985] to suit the autonomic SOA framework in this research is adapted and enhanced. The following algorithm and derived *safe* and *L1-live* Petri net model (Fig. 4.10) describe the mechanism.

Input:

$snap_i$

$receive(marker)_{j,i}$ $i, j \in number$

Output:

$report(s, C)_i$ $s \in states(A_i)$

$send(m)_{i,j}$ $i, j \in number$, m a message of A

Internal:

$internal\text{-}send(m)_{i,j}$ $i, j \in number$, m a message of A

States:

status ∈ {start, snapping, reported} initially start

snap-state, a state of A_i, initially null

∀j ∈ number:

channel-snapped(j) a Boolean, initially false

send-buffer(j), a FIFO of A messages and markers, initially empty

snap-channel(j), a FIFO of A message, initially empty

Transitions:

$snap_i$

Effect:

if status == start then

snap-state = state of A_i

status = snapping

$\forall j \in$ numbers

add "marker" to send-buffer(j)

$receive("marker")_{j, i}$

Effect:

if status == start then

snap-state = state of A_i

status = snapping

$\forall j \in$ numbers

add "marker" to send-buffer(j)

channel-snapped(j) = true

$send(m)_{i, j}$

Precondition:

m is first on send-buffer(j)

Effect

remove first element of send-buffer(j)

$report(s, c)_i$

Precondition:

status = snapping

$\forall j \in$ numbers: channel-snapped(j) = true

s = snap-state

$\forall j \in$ numbers: c(j) = snap-channel(j)

Effect:

status = reported

internal-send(m)$_{i,j}$ in A_i

Precondition:

As for send(m)$_{i,j}$ in A_i

Effect:

add m to send-buffer(j)

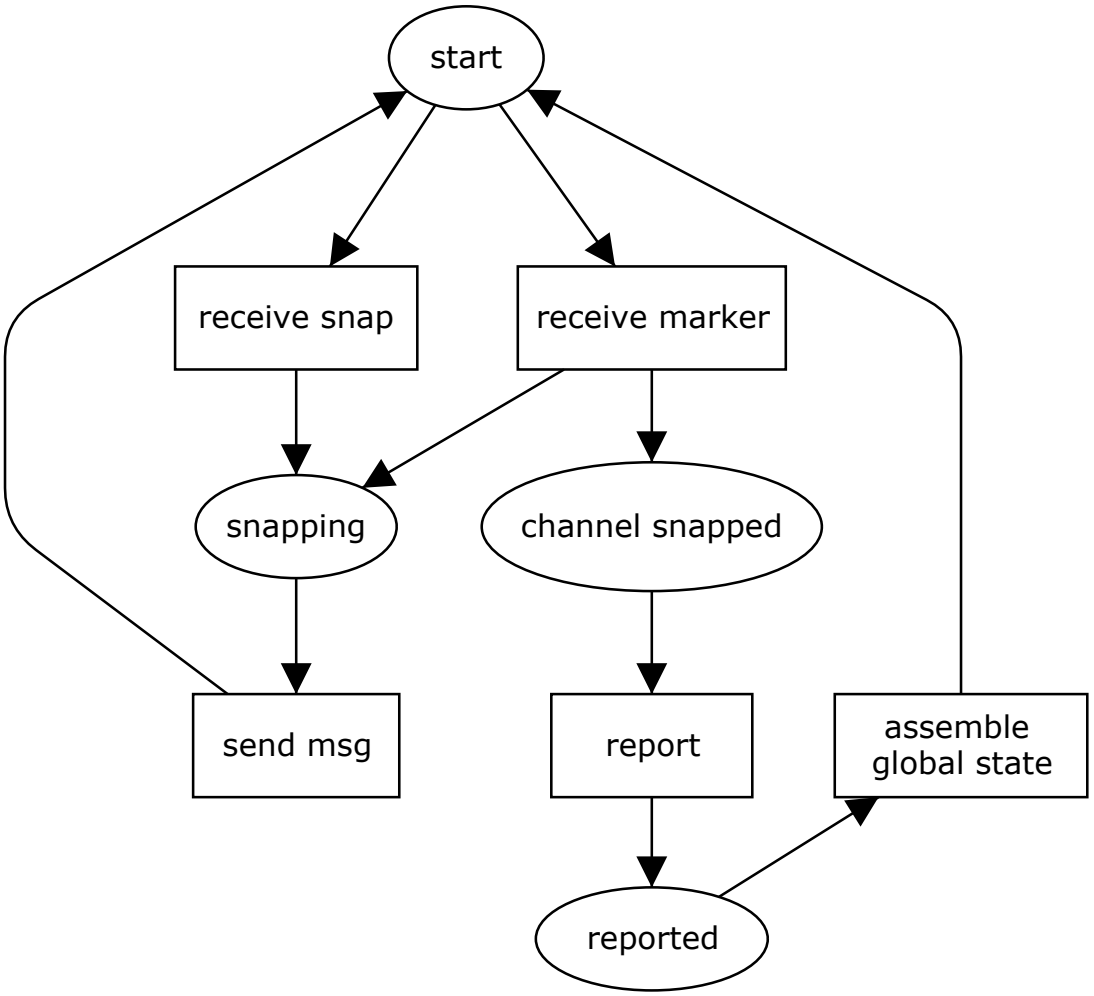

Fig. 4.10 Petri Net model of the snapshot mechanism

The marker receiving and sending rules described by [Chandy & Lamport, 1985] guarantee that if a marker is received along every channel, then each process will record its state and the states of all incoming channels. In particular, if the graph is strongly connected, then all processes will record their states in finite time.

Theorem 1: The snapshot algorithm will determine a global status of autonomic elements (*ae*).

Proof: Once any snap input occurs at autonomic element (ae_i), that element records the state of services in ae_i and sends out markers on all its output channels. Then, when any other autonomic element (ae_j) receives a marker on any channel, it soon records the state of services in ae_j and also sends out markers on all its output channels if it has not previously done so. Due to the strong connectivity of the graph of current internetworking systems, markers will eventually propagate to all autonomic elements that will record their local status. In addition every autonomic element will eventually perform a report output. The recorded process and states will be then collected and assembled to form the recorded global state. As a result, the global status of the autonomic elements is obtained.

4.5 Formal Modeling of Web Services

By using Coloured Petri Nets (CPN) and CPN Tools which provides further insight on the behaviour of the autonomic SOA, especially in situations where actual system testing is not applicable, formal modelling and analysis of the proposed architecture are also conducted.

Details about web services are based on information from WSDL descriptions. Thus to model a web service it is necessary to provide the following WSDL data:

- the name of the web service
- contents of the XML message sent to the external WS (types and names of arguments)
- contents of the response XML message from the external WS (types and names of arguments)
- exceptions for the web service

To invoke a web service and to get a result, the XML messages are used, which contain names and values of input parameters or responses. Meanwhile, to model these XML messages in CPN, appropriate colour sets have to be declared. Record type is used, for enabling mapping names and values as defined in a WSDL description of messages. For example, the web service input (*WS_Input*) and web service output (*WS_Output*) messages are modelled in CPN tools as the following (Fig. 4.11 and Fig. 4.12):

<wsdl:message name="inputMsg">

<wsdl:part name="nameWS" type="xsd:string"/>

<wsdl:part name="inputArg1" type="xsd:int"/>

<wsdl:part name="inputArg2" type="xsd:boolean"/>

</wsdl:message>

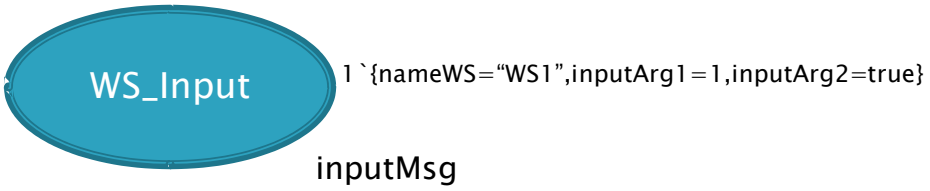

Fig. 4.11 Web service input message in CPN Tools

<wsdl:message name="outputMsg">

<wsdl:part name="nameWS" type="xsd:string"/>

<wsdl:part name="returnType" type="xsd:string"/>

<wsdl:part name="outputArg" type="xsd:int"/>

</wsdl:message>

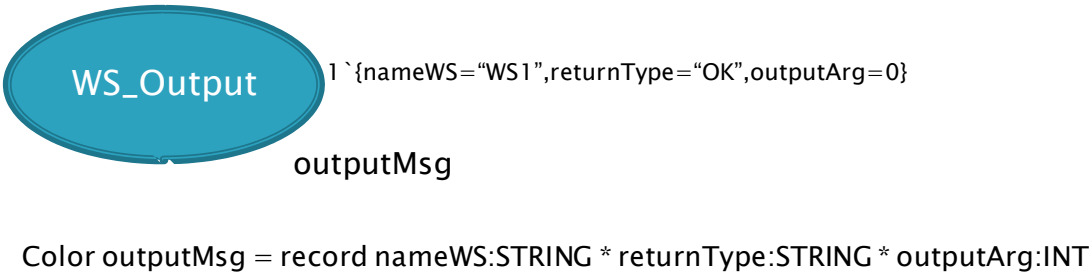

Fig. 4.12 Web service output message in CPN Tools

A web service composition involves three main interactions; namely invoking, sending, and receiving [Zurowska & Deter, 2007]. In the colored Petri nets those interactions are modeled as transitions, thus in this research those three subsets of transitions to represent those operations are derived and enhanced from [Zurowska & Deter, 2007] to cope with exceptional and no response messages, that are: $T_{invokeWS}$, T_{sendWS}, and $T_{receiveWS}$.

A transition *t* that represents an *invoke* operation can be defined as the following:

$t \in T_{invokeWS}$ *iff* $(t \in T) \wedge (size(In(t)) = 1) \wedge (size(Out(t)) >= 2) \wedge (\exists p \in In(t) : C(p) \rightarrow inMsg) \wedge (\exists p1 \in Out(t) : C(p1) \rightarrow outMsg) \wedge (\exists p2 \in Out(t) : C(p2) \rightarrow Revise)$

where:

- *T* is a set of all transitions in a net,
- *In* and *Out* are functions that map a node to its input and output nodes, respectively,
- *size* refers to a size of a set,
- *C* maps a place into its color set,
- $\rightarrow$ maps WS messages into record types,
- *inMsg* and *outMsg* represent accordingly all input and all output messages defined in a WS description for a web service.

The definition shows that a transition modelling an *invoke* operation has one input place with the colour set mapped from a WSDL input message, and at least two output places - one with the colour set mapped from a WSDL output message and

another with the unit colour set (it represents “no response” type of output). The size of the set of output can be bigger than two as in WSDL description it is possible to have fault messages, each of which is modelled as an output place. Fig. 4.13 shows the Petri net model of the invoke operation.

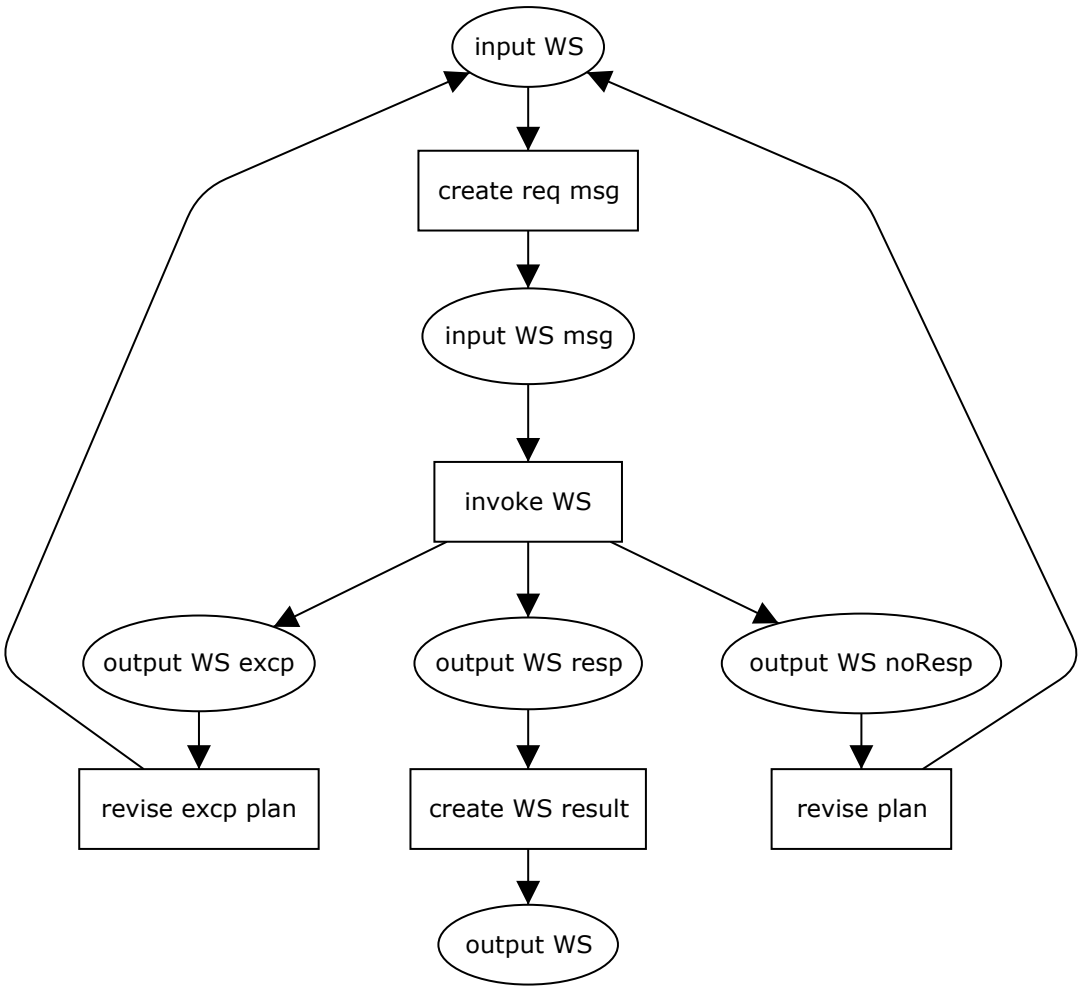

Fig. 4.13 Petri Net model of the invoke operation

A transition *t* that represents a *send* operation can be defined as the following:

$t \in T_{sendWS}$ *iff* $(t \in T) \wedge (size(In(t)) = 1) \wedge (size(Out(t)) = 1) \wedge (\exists p \in In(t) : C(p) \rightarrow inMsg) \wedge (\exists p1 \in Out(t) : C(p1) \rightarrow reqMsg)$

Different from *invoke* operation, in *send* operation there is no any different output type but only the request service message (*reqMsg*) colour set. Fig. 4.14 shows the Petri net model of the *send* operation.

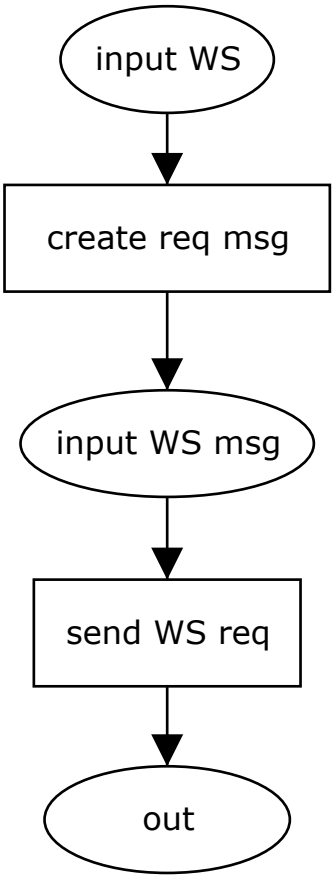

Fig. 4.14 Petri Net model of the send operation

A transition *t* that represents a *receive* operation can be defined as the following:

$t \in T_{receiveWS}$ *iff* $(t \in T) \wedge (size(In(t)) = 1) \wedge (size(Out(t)) >= 2) \wedge (\exists p \in In(t) : C(p) \rightarrow respMsg) \wedge (\exists p1 \in Out(t) : C(p1) \rightarrow outMsg) \wedge (\exists p2 \in Out(t) : C(p2) \rightarrow Revise)$

The difference between this definition and the *invoke* operation is that for input there is the *respMsg* colour set. Thus, an input message is not modelled. Fig. 4.15 shows the Petri net model of the *receive* operation.

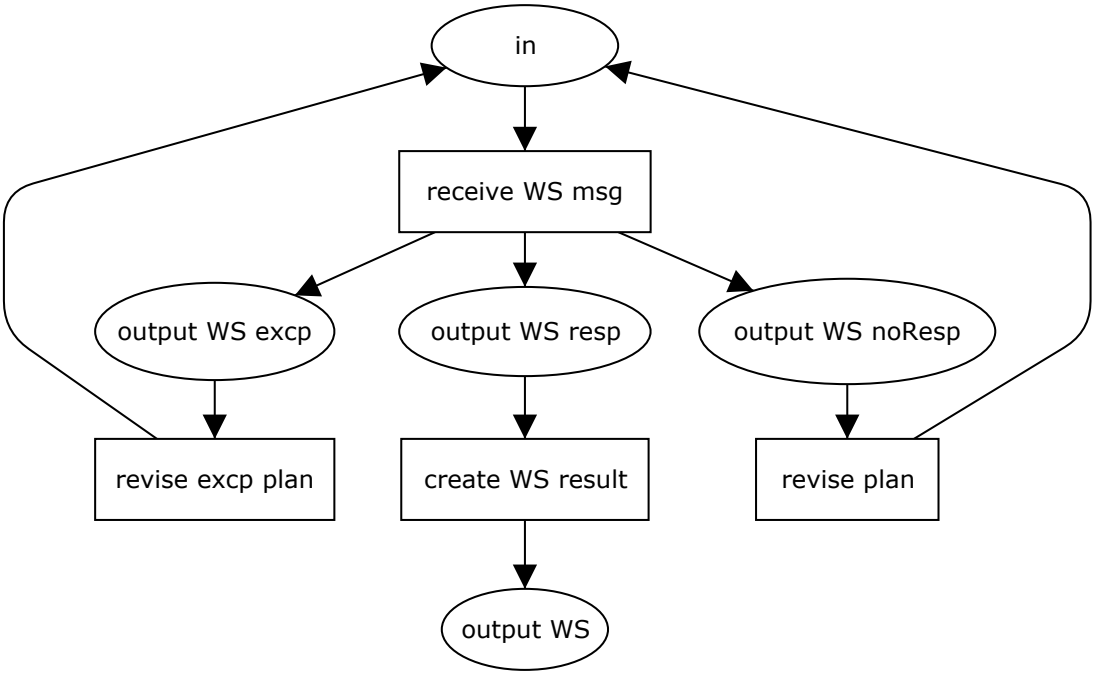

Fig. 4.15 Petri Net model of the receive operation

The set of all interactions for composite web service can be defined as the following:

$$T_{WS} = T_{invokeWS} \cup T_{sendWS} \cup T_{receiveWS}$$

One of the Petri Nets analysis methods are occurrence graphs which in this research are to analyze composite web services to identifying how failures of required web services may influence the overall SOA execution. An occurrence graph is a graph with a node for each reachable marking (a distribution of tokens between places) and an arc for a transition and its binding (called binding elements). This graph is the basis for checking whether composite web service can be successfully executed even if one or more used web services do not respond or give out exceptional message, which is modelled as "*no response*" and "*exceptional*" type of output respectively, and in the colored Petri Nets as output place of an interaction with the unit color set. To perform such checking it is necessary to infer the reachability of a marking representing a success of composite web service

composition from markings representing different outputs from external web services. This analysis was also extending the work by [Zurowska & Deter, 2007].

Occurrence graphs for the objective of this research are a very useful tool. However, a problem emerges that the occurrence graphs may become very large, even for simple composite web service. This is caused by the unpredictability of the actual result of the interactions which, in this research, are modelled with external components. To overcome the problem, the occurrence graphs are used along with equivalence classes (OE-graphs), which can reduce the number of nodes and make the state space analysis more tractable, and the received results from web service are limited into three types of messages: response message, no response, and exceptional message. An equivalence specification is a pair ($\approx_M$, and $\approx_{BE}$), where $\approx_M$ is an equivalence relation on markings and $\approx_{BE}$ is an equivalence relation on binding elements.

In the context of modeling composite web service, those equivalence relations on results from the used web services are defined according to how the results are used in composite web service. Hence it can be assumed that all results from the web service are equal, or several classes of them could be defined as well. The equivalence for markings is:

$$M \approx_M M2 \Rightarrow \forall p \in P : (M_1(p) = M_2(p) \vee (p \in (X(T_{WS}) \cup PN) \wedge M_1(p) \approx_{result} M_2(p)))$$

where:

- P is a set of all places
- PN is a set of port nodes
- T_{WS} is a transition that represents call to an external web service
- X is function that maps a node to a set of its surrounding nodes
- $\approx_{result}$ is an equivalence relation on results from the web service

From the above definition two markings are the same if the only places they differ are the input or output from an interaction page or places surrounding a transition for an interaction with an external web service.

Additionally colours for those places are equal as defined for the web services results. The binding elements equivalence is:

$$BE_1 \approx_{BE} BE_2 \Rightarrow (t(BE_1) = t(BE_2) \wedge (\forall v \in Var(t(BE_1)) (b(BE_1)(v) \approx result\ b(BE_2)(v))))$$

where:

- *t* maps *BE* to its transition
- *b* maps *BE* to its binding
- *Var(t)* is set of variables for transition t
- $\approx result$ is the same as previously

From the definition, two binding elements are equivalent if they are for the same transition and the bindings for variables are equivalent according to the relation defined for the WS results.

For the composite web service, it can be assumed that all results from the web service are equal. To use an OE-graph for checking an influence of failures of other web services on composite web service, it needs to check the reachability of successful execution of composite web service. A marking that represents this state is the one that contains token element only in a place named ”End”, thus *m* is $M_{success}$ iff:

$$((m \in M) \wedge (m(p_{End}) \neq empty) \wedge (\forall p \neq p_{End}\ m(p) = empty))$$

where:

- *M* is a set of all markings in an OE-graph
- p_{End} is a place named “End”

Analogously, the nodes and markings in an OE-graph, that represent exceptional or no response types of output for each used external web service, are identifiable in

the research. Then it is followed by checking the reachability of $M_{success}$ from all those states. If the success is reachable, it enables to execute composite web service even if there is an exception or no response; otherwise in case of a failure of a component, composite web service (in conventional SOA framework) could not be successfully executed. The additional *revise* node makes the proposed framework potentially able to reach $M_{success}$ even in the case where exceptional error message is received from WSn. The framework will revise the composition plan and it will invoke the next web service (WS_{n+1}) instead. Fig. 4.16 shows the occurrence graph with equivalence classes for web service composition, in which the successful marking is represented by node 9. Thus it can be concluded that even if the external web services is not responding or giving exceptional messages, the service composition is likely will still be successful.

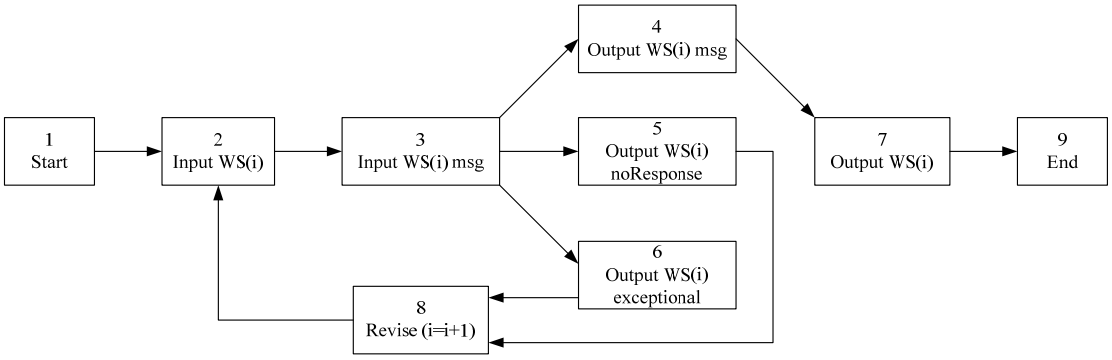

Fig. 4.16 Occurrence graph of web service composition in autonomic SOA

4.6 Chapter Summary

This chapter has elaborated the proposed autonomic SOA framework in detail including its modules, functionalities, processes, and algorithms. The main modules of the architecture are based on autonomic computing cycle, i.e. monitoring, analyzing, planning, and executing, whose functions and algorithms have been described.

The chapter has also described and specified the models of the proposed architecture, including the UML-based meta-model and Petri Net based formal modeling. The Petri Net based models are then used to formally analyze the web services in the autonomic SOA, while the UML models are later used to develop simulation and prototype of the architecture, which will be elaborated in chapter five and chapter six. The Petri Net analysis showed that in the proposed autonomic SOA framework, web service composition will still be successful even if the atomic web services are not responding or giving error messages. The *revise* process makes the proposed framework potentially able to reach successful end marking even in the case where exceptional error message is received from web service provider. The framework will revise the composition plan and it will invoke the next service provider instead.

CHAPTER 5

APPLICATION DOMAIN

5.0 Chapter Overview

The benefits of this research can be applied to many applications that make use of service-oriented architecture and service ecosystems, such as business applications, health care systems, bioinformatics, telecommunication services, and travelling services. This chapter, divided into two main sections, presents some case studies as illustrative examples to give impression of the target research area and possible application domain of the research. These case studies will show the feasibility of implementing the proposed framework of this research in many application domains and how will they benefit from the proposed framework. The first section provides an overview on some of the available real-world applications that will benefit from the proposed framework, while the second section thoroughly discusses selected case study.

5.1 SOA Application Domain

This section presents some example of real-world applications that make use of service oriented architecture that would benefit from the proposed work. These applications include mobile telecommunication commerce and medical informatics applications.

5.1.1 Mobile Commerce Application

Service oriented architecture application in telecommunication will focus on a development of integrated and advanced telecommunications services for advanced mobile service. Such services combine several value-added application capabilities with internet and next generation mobile telecommunication capabilities. All these capabilities can be integrated by the autonomic SOA to provide combinations of call/session control, messaging features, presence and location features, single and multiplayer gaming, multimedia content steaming, parental monitoring, accounting and billing, etc.

The environment is particularly challenging, because the network infrastructure and many of the applications that provide the service components are owned and managed by different enterprises (i.e., the network operators / providers, third-party service providers, banks, etc). Furthermore, the environment might be changing over time due to addition, removals, and changes of providers or operators.

Fig. 5.1 illustrates an example configuration of the mobile commerce (m-commerce) system. The use of autonomic and self-organizing SOA should help to provide flexible service collaborations between telecommunication network providers and third parties service providers. Conventional SOA will have difficulties in maintaining service level to customers in times of unpredicted behaviours and failures of the system or service providers. The autonomic manager should be implemented at network service provider's (network operator) site. With the case-based reasoning (CBR) and snapshot mechanism, the autonomic service architecture will be able to cope with those unpredicted behaviours by identifying the status of cooperating service providers and access other providers (within the same network operator) when a service is unavailable or no response message received.

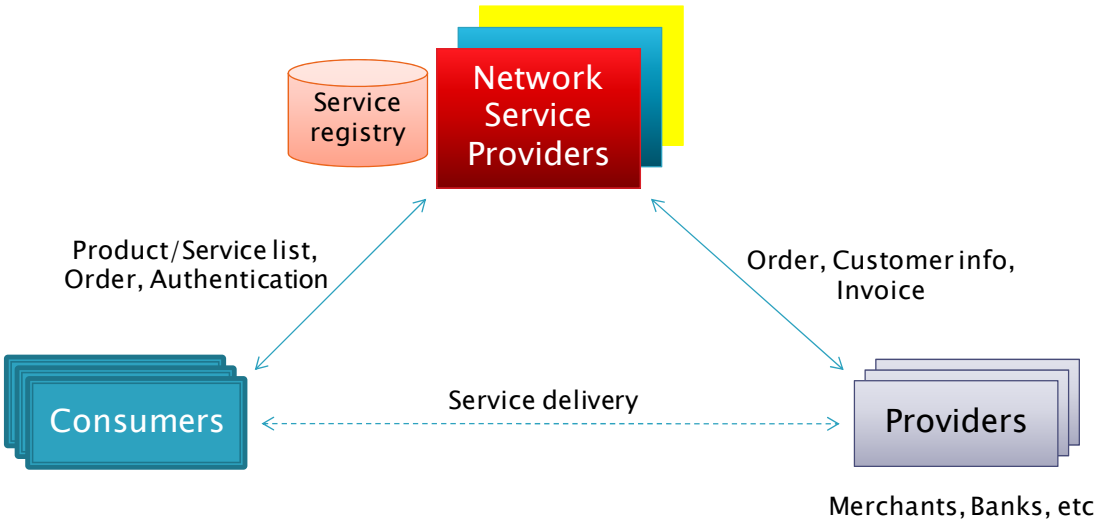

Fig. 5.1 Example configuration of m-commerce

The architecture could also be extended to enable collaboration between network operators when providing services to other operator's subscribers. In order to provide a scalable framework, it needs to extend the ecosystem to other service ecosystems that may be owned by different operators / providers and this can be supported by the framework provided by [Yelmo et al., 2009]. It will enable collaboration between service ecosystems – to create a bigger service ecosystem – when providing services to users, using collaboration agreements. The framework was designed for mobile operators in telecommunication service ecosystems and it is likely that the framework is also able to be adapted into other types of service ecosystems.

Fig. 5.2 illustrates a collaborative framework for the autonomic SOA. The service ecosystem with autonomic SOA will be able to collaborate with other web service ecosystems using the collaboration agreements framework, managed by the autonomic manager.

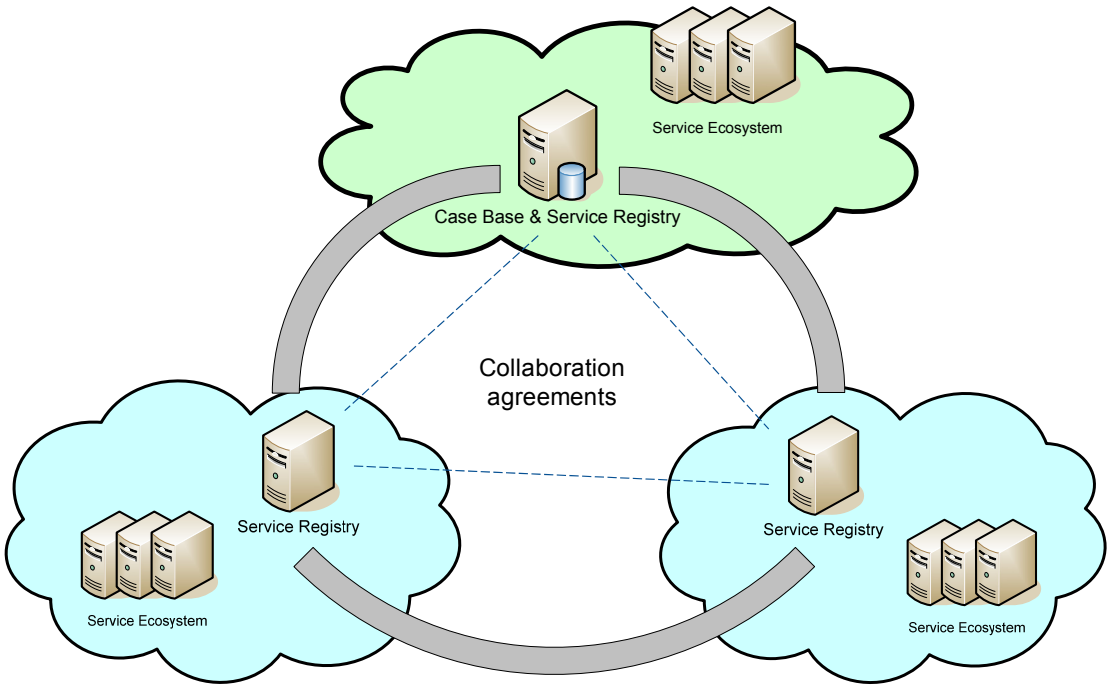

Fig. 5.2 Adaptation of collaborative framework in autonomic SOA

5.1.2 Healthcare Informatics Application

Studies on SOA for healthcare have been given by [Juneja et al., 2008], [Juneja et al., 2009], [Daskalakis & Mantas, 2009], and [Smith & Lewis, 2009] to address healthcare service integration with SOA. Several constraining factors in healthcare industry could include lack of funding, challenges in achieving regulatory compliance, fragmentation in healthcare industry, strongly hierarchical decision-making within organizations, extensive needs for security and difficulty in reaching consensus on shared data. These factors point out that the industry has a number of unique business needs, such as a unique set of business processes and data, a heavy regulatory environment and different sets of stakeholders with frequently conflicting needs and goals. However, the health industry also confronts a set of IT problems similar to many industries, such as defining and modelling essential business information and business rules, storing and accessing information in support of business processes, and assuring the security, performance, availability and usability of IT systems.

[Smith & Lewis, 2009] believed that SOA can enable business agility, leverage of legacy investments, adaptability and cost-efficiency all of which support the goal of developing effective healthcare information systems. SOA adoption has the potential of providing real value for healthcare organizations to realize benefits such as cost-efficiency, adaptability, leverage of legacy systems, and the business agility required to meet new healthcare needs. Fig. 5.3 illustrates an example of a service-oriented system in healthcare domain given by [Smith & Lewis, 2009].

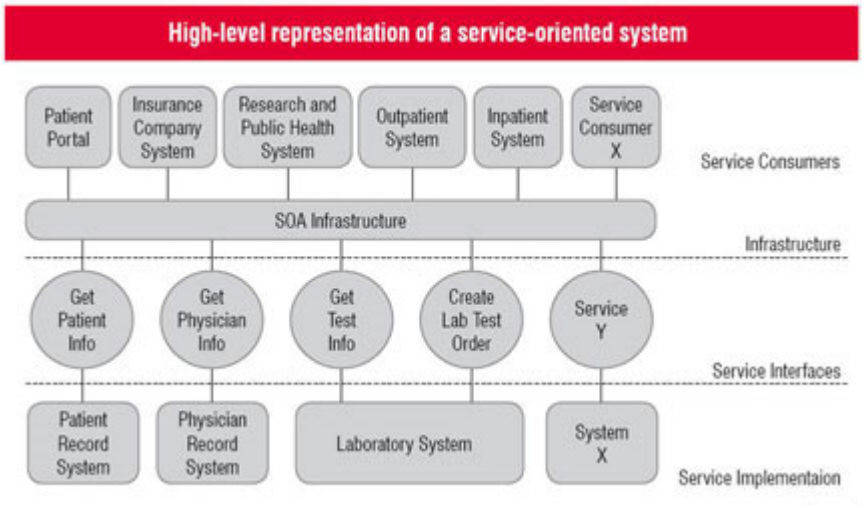

Fig. 5.3 Example of SOA model in healthcare system [Smith & Lewis, 2009]

Services are reusable components that represent business tasks, such as:

- Patient lookup
- Patient medical history lookup
- Medical lab test order
- Insurance lookup

Services can be globally distributed across organizations and support a number of business processes. Service consumers use the functionality provided by the services. Some examples of service consumers are end-user applications, health information network portals, and internal and external systems. SOA infrastructure here could function to connect service consumers to services through an agreed upon communication model. It often contains elements to support service discovery, security, data transformation, and other operations.

The healthcare system environment in its full-scale implementation, involving the ownership and management of many various hospitals, clinics, laboratories, pharmacies, insurance companies, etc for both service and network infrastructure, equal to the mobile commerce presented previously, can also be quite challenging. Fig. 5.4 shows an example of healthcare information networks SOA.

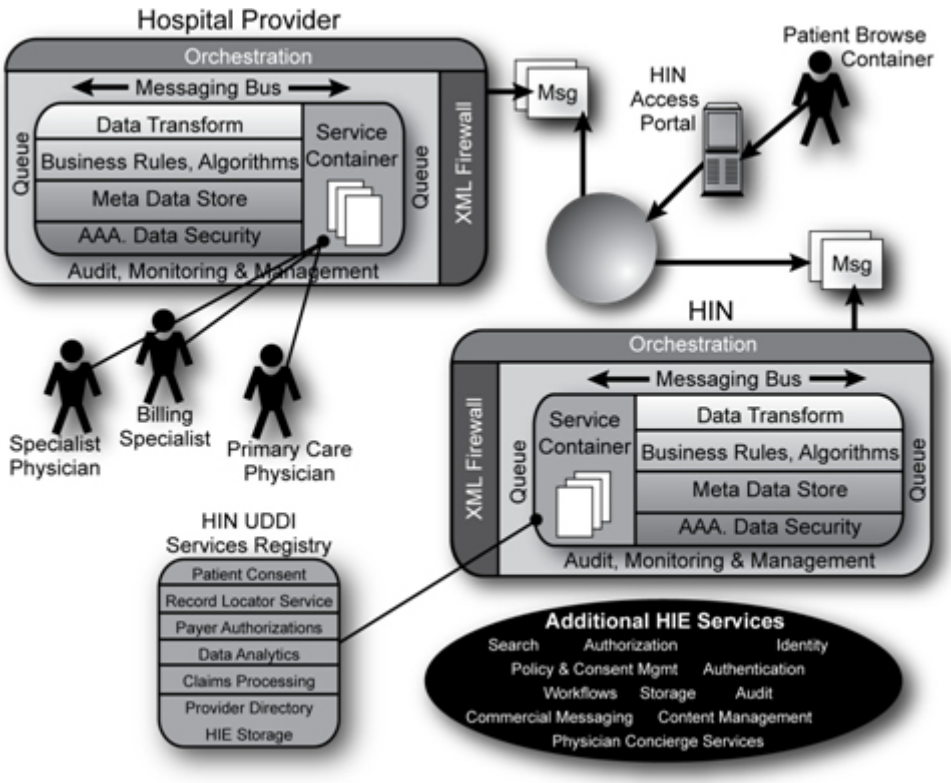

Fig. 5.4 Possible healthcare information network SOA [Juneja et al., 2009]

The application of SOA in healthcare systems can substantially reduce the complexity and redundant system processing of clinical information. It can also help to simplify and reduce the cost of participation in community of care health information networks, and can improve the cost and usability of electronic medical records, and also availability can be increased through service redundancy. The addition of autonomic feature into the business rules and algorithms layer in the architecture shown above would then help to provide adaptive health service collaboration between parties and further improve the overall usability and

availability of the healthcare information networks. Conventional SOA applications will have difficulties in maintaining service level to customers in times of unpredicted behaviours and failures of the system or service providers. With the autonomic computing paradigm, the autonomic SOA will be able to cope with those unpredicted behaviours by seamlessly accessing other health service providers when a service provider is unavailable, or no response received.

5.2 Selected Application Domain

This section presents two selected case studies to show the feasibility of implementing the proposed framework and its advantages over the conventional SOA framework. The case studies are currency converter and vacation / travel planner services.

5.2.1 Currency Converter Service

A test environment whose goal is to show the ability of the proposed framework to cope with unavailable services was developed by using the following WSDL files:

1. *Currency Convertor* web service [Currency Convertor]
2. *Currency Service* web service [Currency Service]

This case study will show the ability of the autonomic SOA framework to cope with erroneous or unavailable atomic services. These currency converter services were selected as they are freely available on the internet, and they provide the equal atomic service, i.e. providing conversion rate for a given two currencies.

5.2.1.1 Simulation Development

To simulate and test the framework proposed in this research, soapUI, a Java-based free and open source cross-platform testing solution for SOA, is used. Equipped with a graphical interface, and enterprise-class features, soapUI allows users to create and execute automated functional, regression, compliance, and load tests. In a single test

environment, soapUI provides complete test coverage and supports all standard protocols and technologies, including SOAP and REST-based Web services, JMS enterprise messaging layers, databases, and Rich Internet Applications.

Fig. 5.5 and Fig. 5.6 show the two currency converter web services added to the soapUI simulation environment.

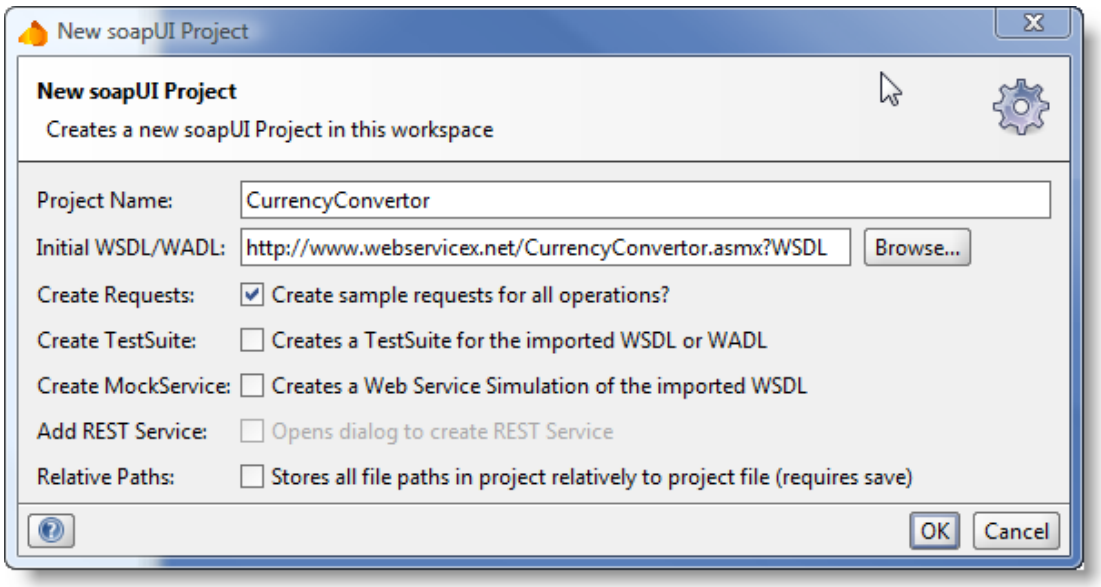

Fig. 5.5 Currency Convertor project using the first web service (*CurrencyCovertor*)

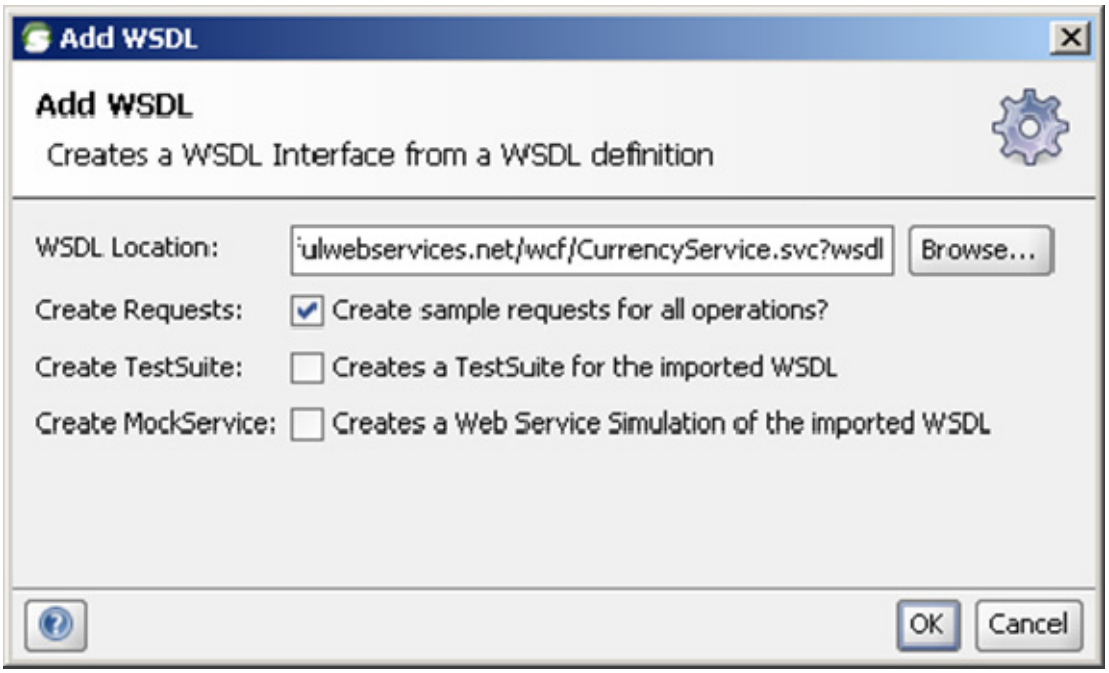

Fig. 5.6 The second web service (*CurrencyService*) is added to the simulation

It is then followed by creating the mock services of those services. Mock services can be used to create a proof of concept, either as a wire frame or as a demo for the proposed framework. This is a powerful means and provides a good ground for decision-making of the framework. Fig. 5.7 shows the overview of the simulation project. There are three service interfaces instead of two because the *CurrencyConvertor* service has two interfaces, one for SOAP 1.1 (*CurrencyConvertorSoap*) and the other one for SOAP 1.2 (*CurrencyConvertorSoap12*).

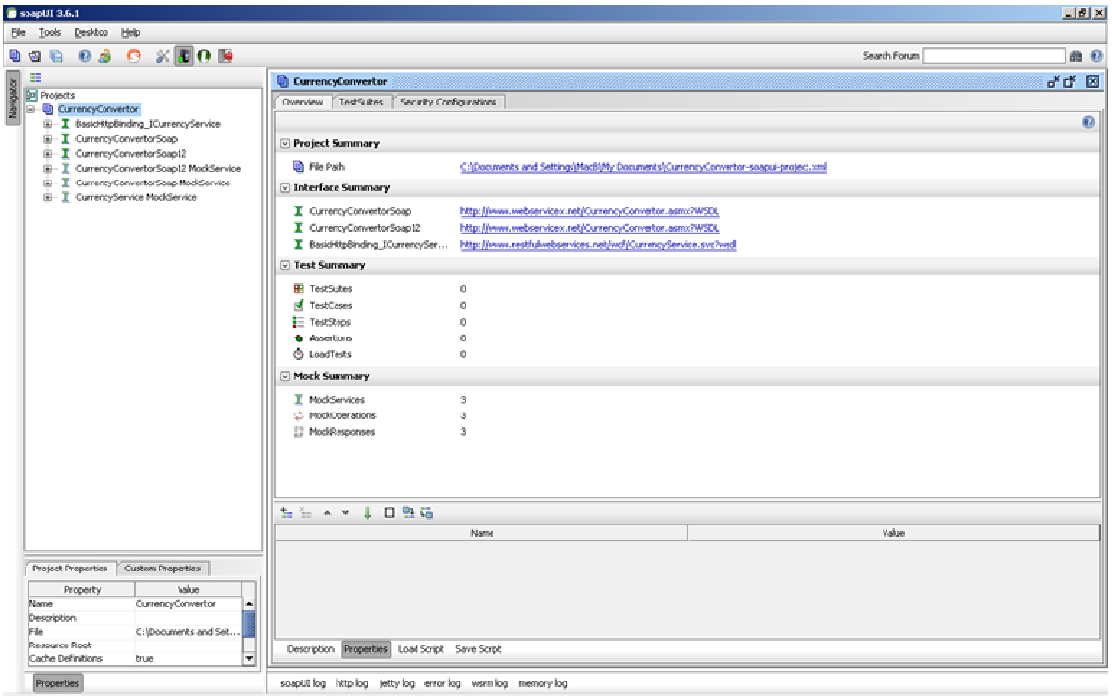

Fig. 5.7 Simulation project overview in soapUI

5.2.1.2 Simulation Results and Analysis

The simulation program was executed several times for the following conversion:

- US Dollar (USD) to Malaysia Ringgit (MYR)
- Euro (EUR) to Malaysia Ringgit (MYR)
- Malaysia Ringgit (MYR) to Indonesia Rupiah (IDR)

The web service providers were simulated to be down (unavailable) alternatingly. Table 5.1 shows the currency converter simulation results.

Table 5.1 Currency converter simulation results

Conversion (from, to)	***CurrencyConvertor***	***CurrencyService***	**Conversion Result (Reachable)?**
USD, MYR	Output message	No response	Yes
USD, MYR	No response	Output message	Yes
EUR, MYR	Output message	No response	Yes
EUR, MYR	No response	Output message	Yes
MYR, IDR	Output message	No response	Yes
MYR, IDR	No response	Output message	Yes

The results showed that the proposed autonomic SOA framework was able to keep providing conversion rate service to the user every time. The autonomic SOA will seamlessly switch and access the *CurrencyConvertor* when *CurrencyService* was unavailable and vice versa, thus increasing the overall system robustness and reliability.

Fig. 5.8 shows a screenshot of erroneous web service in the simulation, i.e. *CurrencyConvertor* service. In this example, the socket time out exection message was displayed after the system tried for some times to connect to the web service. Without the autonomic feature activated, the simulation stopped and user must create new request to try to re-connect or try other service provider.

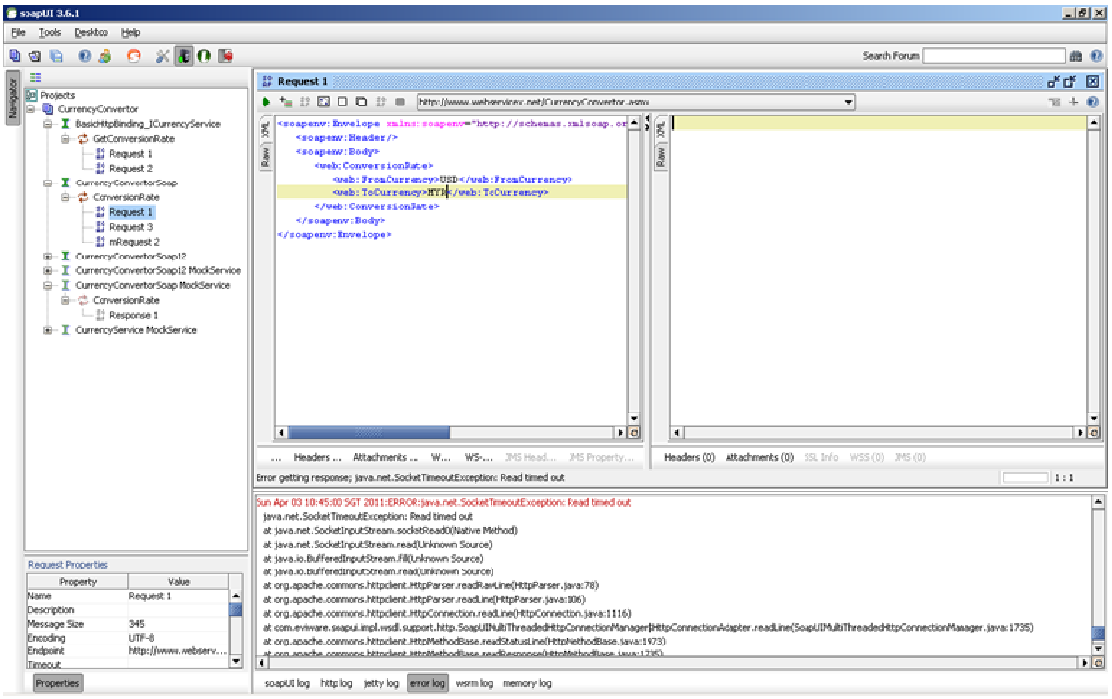

Fig. 5.8 Example of erroneous *CurrencyConvertor* service

However, with the autonomic feature activated, when the *CurrencyConvertor* web service was unavailable, the system was still able to provide the conversion rate by seamlessly switch to the other service provider, i.e. *CurrencyService* as shown in Fig. 5.9. Other snapshots and code of the simulation work are also provided in Appendix A and Appendix B.

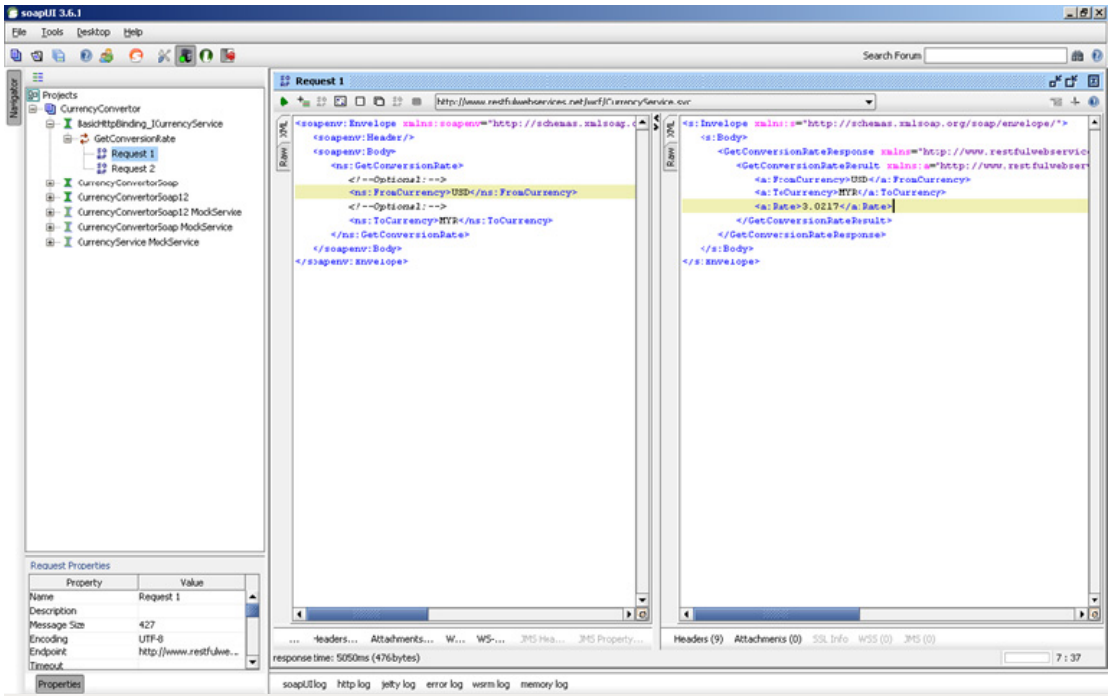

Fig. 5.9 The proposed framework seamlessly switch to *CurrencyService*

5.2.2 Travel / Vacation Planner

To compare the research as peer-to-peer, the following works that also used Petri Nets modeling are chosen and presented. This case study will show the ability of the proposed autonomic SOA framework to cope with unavailable services in service composition.

5.2.2.1 Travel Scheduling

[Yoo et al., 2009] used travel scheduling as a case study. The state diagram of travel scheduling is given in Fig. 5.10 and Fig. 5.11 shows the Petri Nets model of the travel scheduling.

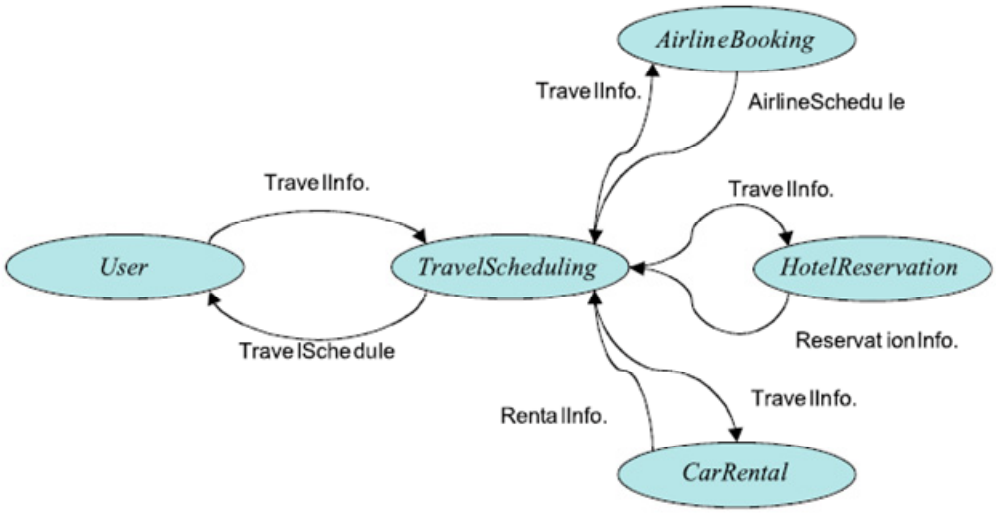

Fig. 5.10 State diagram for travel scheduling [Yoo et al., 2009]

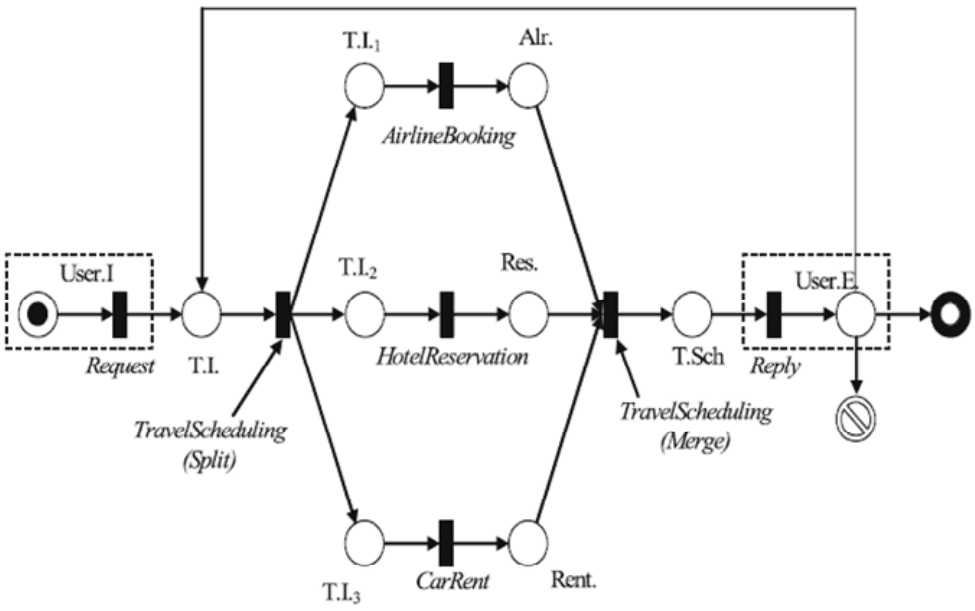

Fig. 5.11 Petri Nets model for travel scheduling [Yoo et al., 2009]

The conditions of their work are the following:

The validation conditions:

- Visit = AirlineBooking & HotelReservation & CarRental
- Initial input = TravelInfo

- Final output = TravelSchedule
- Final status = Success (Accept)|Failure (Reject)

Place = [User.I, T.I, T.I1, T.I2, T.I.3, Alr, Res, Rent, T.Sch, User.E].
Transition = [Request, TravelScheduling.S, AirlineBooking, HotelReservation, CarRent, TravleScheduling.M, Reply].

Their results included the reachability tree of the travel scheduling process as shown in Fig. 5.12.

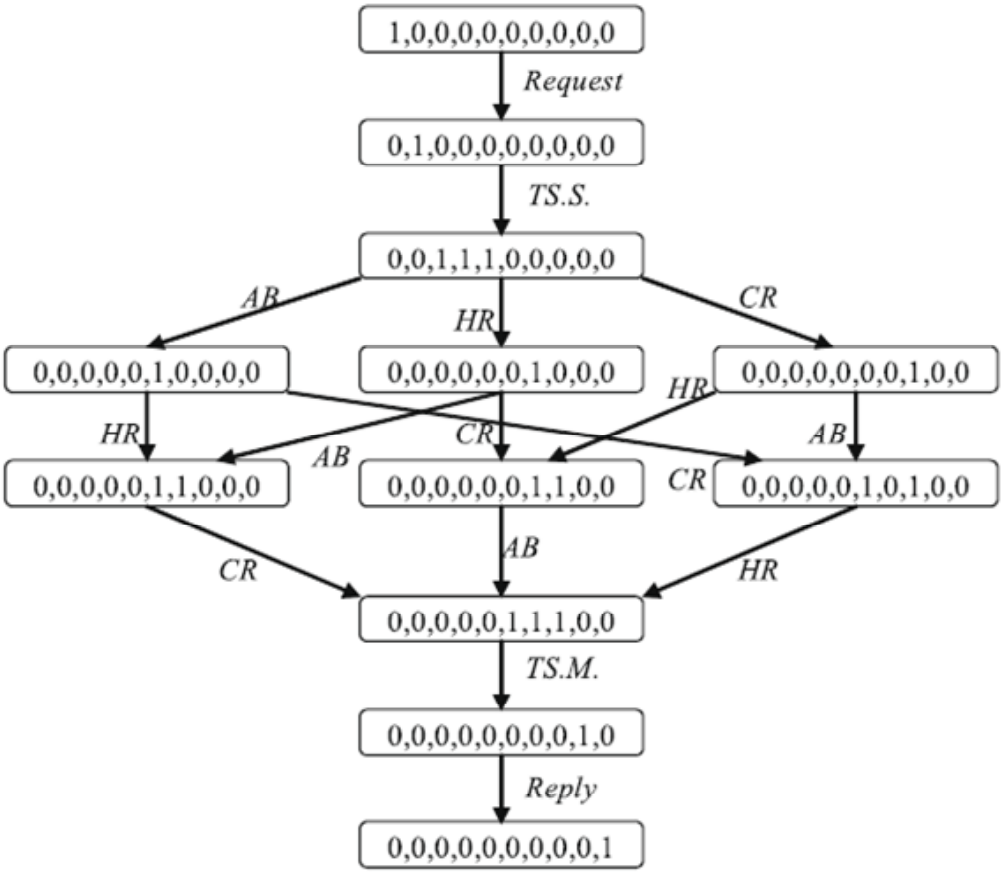

Fig. 5.12 Reachability for travel scheduling [Yoo et al., 2009]

Their results stated that "the service composition is complete and logically correct if no exception / error occurs from the (initiating) user to the (terminating) user" [Yoo et al, 2009]. In this aspect, the autonomic SOA framework in this research is better for

being able to cope for non responsive atomic services, exceptions and errors messages happened in service composition as shown in the formal models and analysis in section 4.1.7.

If any error or exception is raised in service composition, it will be captured by the monitoring module and the CBR process will analyze the error and plan action to overcome the error accordingly. The action plan may include usage of other service provider (in case of web service provide error or unavailability) or usage of other channel of communication (in case of network problem).

5.2.2.2 Vacation Planner

[Zurowska and Deter, 2007] used vacation planner as a case study in their work. Fig. 5.13 shows the CPN model of the vacation planner.

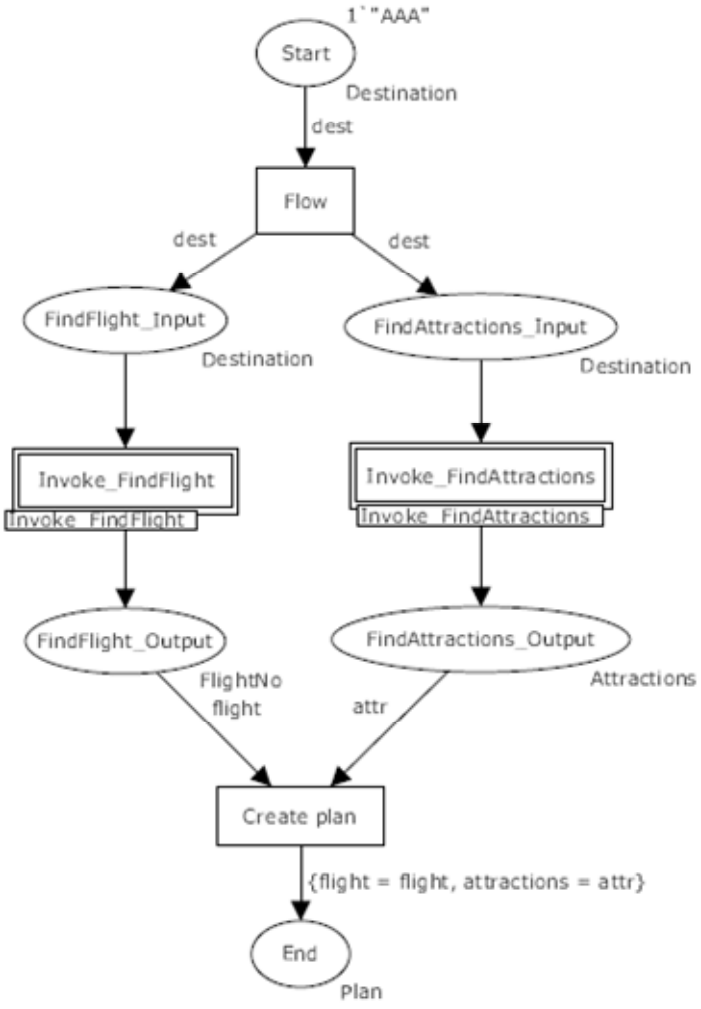

Fig. 5.13 CPN model for vacation planner [Zurowska & Deter, 2007]

Their result showed that their framework was able to void interactions with optional components (web services) that are not working. However, in the case when the faulty web service is compulsory to successfully execute composite web service (like *FindFlight* in their example, shown in Fig. 5.13), the system was unable to overcome it. This is shown in their reachability analysis in Table 5.2.

Table 5.2 Reachability analysis for the vacation planner [Zurowska & Deter, 2007]

"From" marking	"To" marking	Reachable
Output_FindAttractions_No + Output_FindFlight_Msg	End	Yes
Output_FindAttractions_Msg + Output_FindFlight_No	End	No

In their case study, the *FindFlight* web service is a compulsory service and the *FindAttractions* web service is an optional one. If there is a valid output message from the *FindFlight* web service and no valid output from the *FindAttractions* web service, the end state is still reachable. However if there is no valid output from the *FindFlight*, even if there is a valid output from *FindAttractions*, the end state will be unreachable.

This case study was simulated using the WSDL descriptions given by [Zurowska & Deter, 2007] in soapUI environment. Table 5.3 shows simulation results of the vacation planner in autonomic SOA.

Table 5.3 Vacation planner simulation results

FindAttractions	***FindFlight***	**Vacation Booking Result (Reachable)?**
No response	Output message	Yes
Output message	No response	Yes

From this aspect, the proposed autonomic SOA framework of this research is also better compared to the conventional SOA framework analyzed by [Zurowska & Deter, 2007], because it is still able to reach success end state ($M_{success}$) even if there is no valid output from *FindFlight* (no response or exceptional message) as shown in the simulation results and also described in the formal analysis in chapter four. This is true due to the ability of the framework to revise its action plan and look for other services similar to what *FindFlight* provides, either within the service ecosystem (via the snapshot algorithm) or searching at other service ecosystems (based on collaborative agreements).

5.3 Chapter Summary

This chapter has presented several case studies in which the architecture proposed in this research can be applied. In general, these case studies showed the feasibility of implementing the proposed autonomous service architecture in real world applications and illustrated how those applications will benefit from the proposed framework.

Furthermore, the provided discussion and analysis on two of the case studies, i.e. currency converter and travel / vacation planner, showed the advantages of the proposed autonomous service architecture compared to conventional SOA framework. The simulation results showed the ability of the proposed framework to work around unavailable services and seamlessly provide user with the same type of service from different service providers. Therefore the framework will improve the success rate of providing not only atomic service, but also composite service since it improves the availability and reliability of the atomic services.

CHAPTER 6

PROTOTYPE DESIGN AND DEVELOPMENT

6.0 Chapter Overview

Prototype design and development the proposed architecture are presented in this chapter. The design and development experiences and lessons learned from applying the proposed framework are then discussed.

6.1 Prototype Design

In this research an effort to design and develop prototype of the proposed autonomic SOA for the computational engineering case study using Java-based platform Apache Axis2 [Apache-Axis] as the web services / SOAP engine in Windows XP-based computer has been started. In its process, some applications via web services (e.g. SunFlow [SunFlow] rendering system and survey data visualizer) have also been integrated.

Engineers and researchers in computational engineering typically use different types of software and applications in their workflow. The applications might include modelling, simulation, and visualization software. The software environments sometimes are complex and dynamic, supporting different software packages, codes, and possibly distributed in different locations. The use of SOA in turn will help to integrate the different applications/software used in computational engineering.

Fig. 6.1 illustrates the project that has been initiated and aimed to provide an integrated services framework for various distributed software used in computational engineering research, in-house software and commercial software, including modelling software, simulation software, and visualization software. The modelling software will include computational software using Ray tracing technique and Finite-difference Time-domain (FDTD) method. The simulation software will run on High Performance Computing (HPC) platform. Further, the visualization software will include 2D visualization (graphs and charts), 3D visualization, and virtual reality visualization tools.

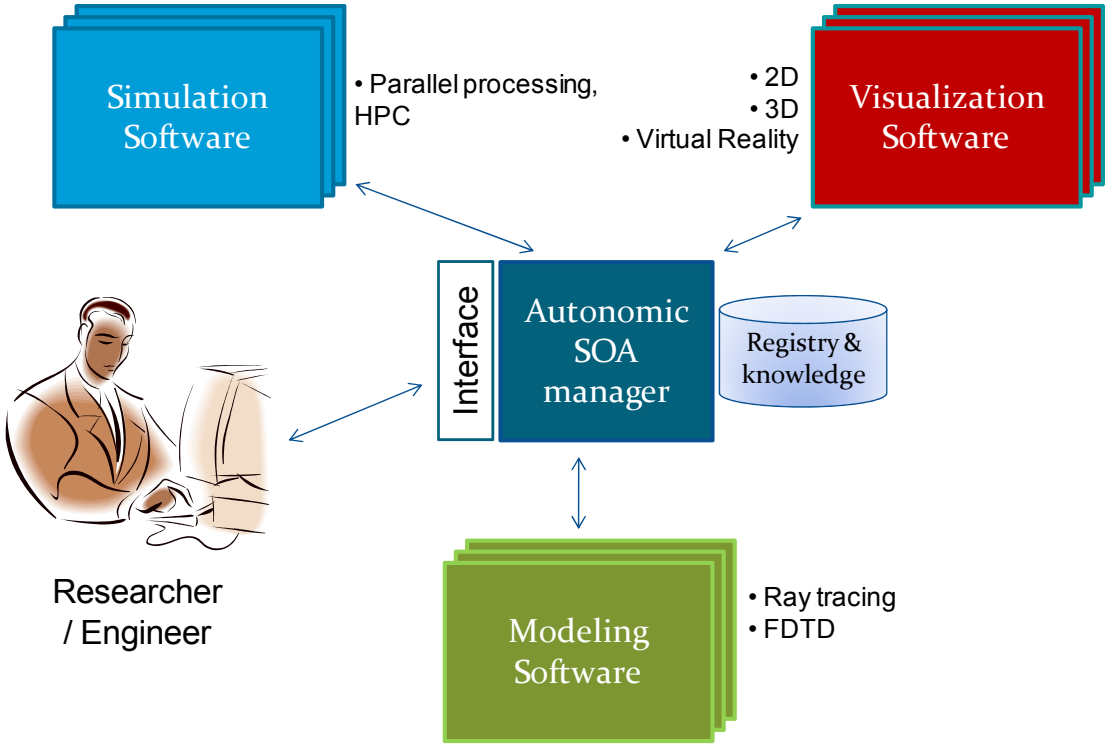

Fig. 6.1 Implementation model of the computational engineering project

This project is able to be implemented using conventional SOA framework. For example, the work by [Kim et al., 2006] described a construction of a SOA in engineering framework. As a case study, they used an engineering process for a pump design using several engineering software, distributed at different laboratories.

However it is believed that there will be limitations in term of flexibility and adaptability in using conventional SOA approach. In engineering research, which implementation software intended to be used cannot be normally fixed. For example, some engineers at the moment may be using a commercial FDTD version. But later, they may switch to in-house built version, and a couple of weeks later to a freeware version. Over the course of the project, an autonomic manager will be able to suggest to the users (engineers) which software to use and, by doing so, the users further will develop a sense of which software implementation is most suitable for certain specific purpose. This feature could not be achieved using conventional SOA approach.

Fig. 6.2 shows an instance of computational engineering process in choreography. For example, the system could use ray-tracing and FDTD modelling software, and virtual reality software for visualization. It is shown that some actions are initiated by user and others are by the autonomic manager automatically. The autonomic manager will appropriately learn and adapt the sequences of tasks to solve computational engineering problems based on the previous cases stored in the knowledge base.

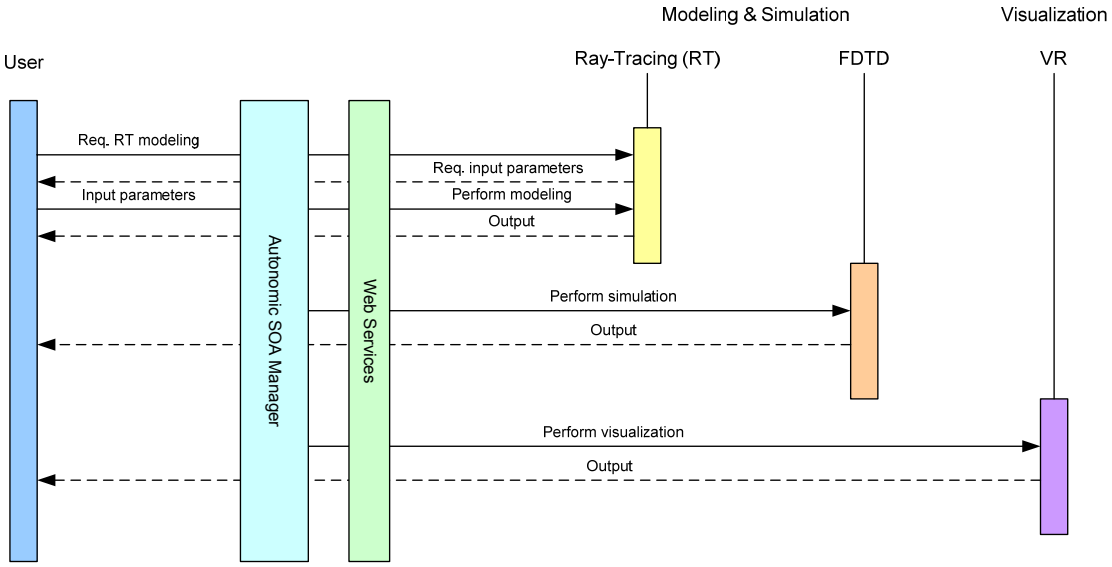

Fig. 6.2 Example of interaction model in computational engineering process

Software specifications:

- Operating System: Windows XP
- Java Technology: Java Development Kit Version 6 - http://java.sun.com/javase/downloads/index.jsp
- Web Server: Apache Tomcat Version 5.5 - http://tomcat.apache.org/download-55.cgi
- Database Server: MySQL Server Version 5 - http://dev.mysql.com/downloads/mysql/
- JDBC Driver – Connector / J Version 5.1.12 - http://dev.mysql.com/downloads/connector/j/
- Web Browser: IE 8 and Mozilla Firefox 3.6 - http://www.microsoft.com/nz/windows/internet-explorer/default.aspx
- SOA Engine: Apache Axis2 1.4.1 Standard Distribution and war - http://ws.apache.org/axis2/download/1_4_1/download.cgi

6.2 Database Design

Table 6.1 through Table 6.5 show the design and structure of the database, and Fig. 6.3 and Fig. 6.4 show snapshots of the tables in database. Other snapshots and code of the prototype work are provided in Appendix A and Appendix B.

- Table Name: **Application_Details**
  PrimaryKey: **Application_Id**

Table 6.1 Application_Details table

Column	Data Type	Description
Application_Name	Varchar(15)	It stores name of the Application
Application_Type	Varchar(15)	It stores about the type of the Application
Application_Id	Int	It stores the id of the Application which will be a Primary Key
Application_Reg_Status	Varchar(1)	It stores whether Application is Registered or not
Application_Service_Status	Varchar(1)	It stores whether the Application is Available or not
Application_Count	Int	It stores the total no of Applications which are registered

- Table Name: **Modelling**
  PrimaryKey: **Mod_App_Id, Mod_App_Name**

Table 6.2 Modeling table

Column	Data Type	Description
Mod_App_Id	Int	It stores the id of the Application and it will be part of the Primary Key
Mod_App_Name	Varchar(15)	It stores name of the Modelling Application and it will be part of the Primary Key
Url	Varchar(50)	It stores the URL of the Application
Description	Varchar(100)	It stores short description about the Application
No_of_Mod_App	Int	It stores the total no of Modelling Applications
Mod_App_Lastuptime	DateTime	It stores the recent Uptime of the Modelling Application
Mod_App_Lastdowntime	DateTime	It stores the recent Downtime of the Modelling Application

- Table Name: **Simulation**
  PrimaryKey: **Sim_App_Id, Sim_App_Name**

Table 6.3 Simulation table

Column	Data Type	Description
Sim_App_Id	Int	It stores the id of the Application and it will be part of the Primary Key
Sim_App_Name	Varchar(15)	It stores name of the Simulation Application and it will be part of the Primary Key
Url	Varchar(50)	It stores the URL of the Application
Description	Varchar(100)	It stores short description about the Application
No_of_Sim_App	Int	It stores the total no of Simulation Applications
Sim_App_Lastuptime	DateTime	It stores the recent Uptime of the Simulation Application
Sim_App_Lastdowntime	DateTime	It stores the recent Downtime of the Simulation Application

- Table Name: **Visualisation**

  PrimaryKey: **Vis_App_Id, Vis_App_Name**

Table 6.4 Visualization table

Column	Data Type	Description
Vis_App_Id	Int	It stores the id of the Application and it will be part of the Primary Key
Vis_App_Name	Varchar(15)	It stores name of the Visualisation Application and it will be part of the Primary Key
Url	Varchar(50)	It stores the URL of the Application
Description	Varchar(100)	It stores short description about the Application
No_of_Vis_App	Int	It stores the total no of Visualisation Applications
Vis_App_Lastuptime	DateTime	It stores the recent Uptime of the Visualisation Application
Vis_App_Lastdowntime	DateTime	It stores the recent Downtime of the Visualisation Application

- Table Name: **Service_History**
  PrimaryKey: **Service_Name**

Table 6.5 Service_History table

Column	Data Type	Description
Service_Name	Varchar(15)	It stores the name of the Service.
Service_Id	Int	It stores the id of the Service
Serviced_Count	Int	It stores the no of times the service has been Serviced.
Service_Type	Varchar(10)	It stores whether the Service is Composite or Atomic

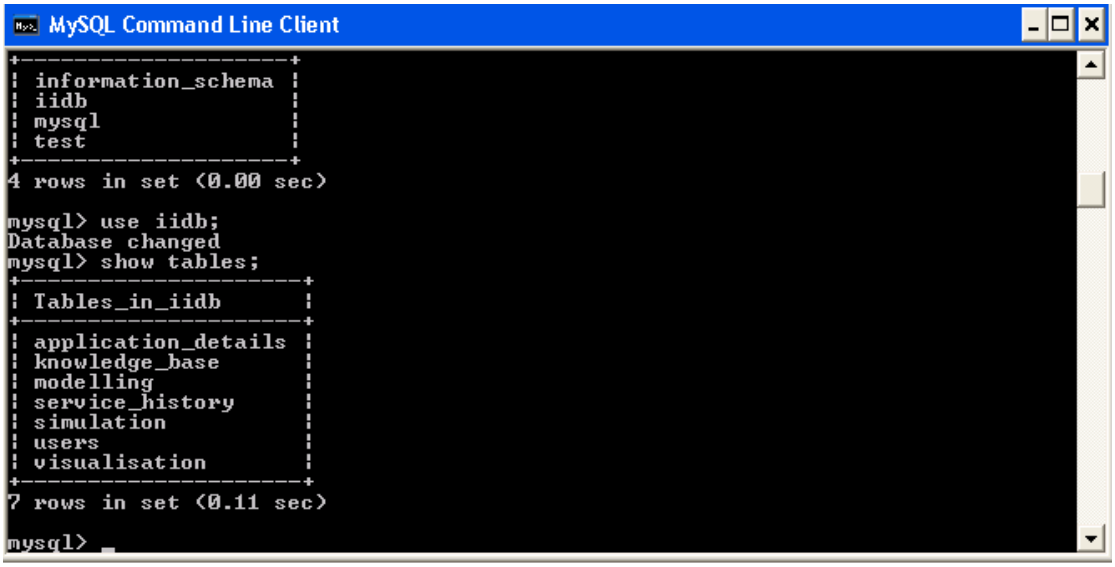

Fig. 6.3 Snapshot of tables in the database

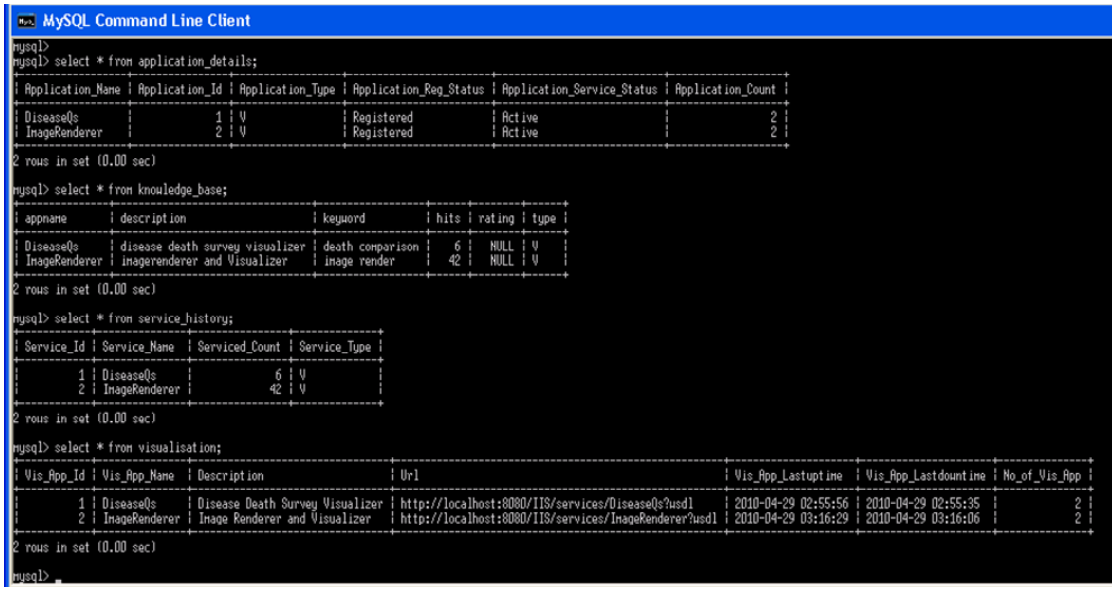

Fig. 6.4 Snapshot of tables contents in database

6.3 Class Diagrams

Packages:

- Service Consumer Package: containing objects requesting for services.
- Service Register: containing objects used for registering, identifying and discovering the services.
- Service Producer: containing Objects providing services.

Fig. 6.5 through 6.9 shows the diagrams of the packages.

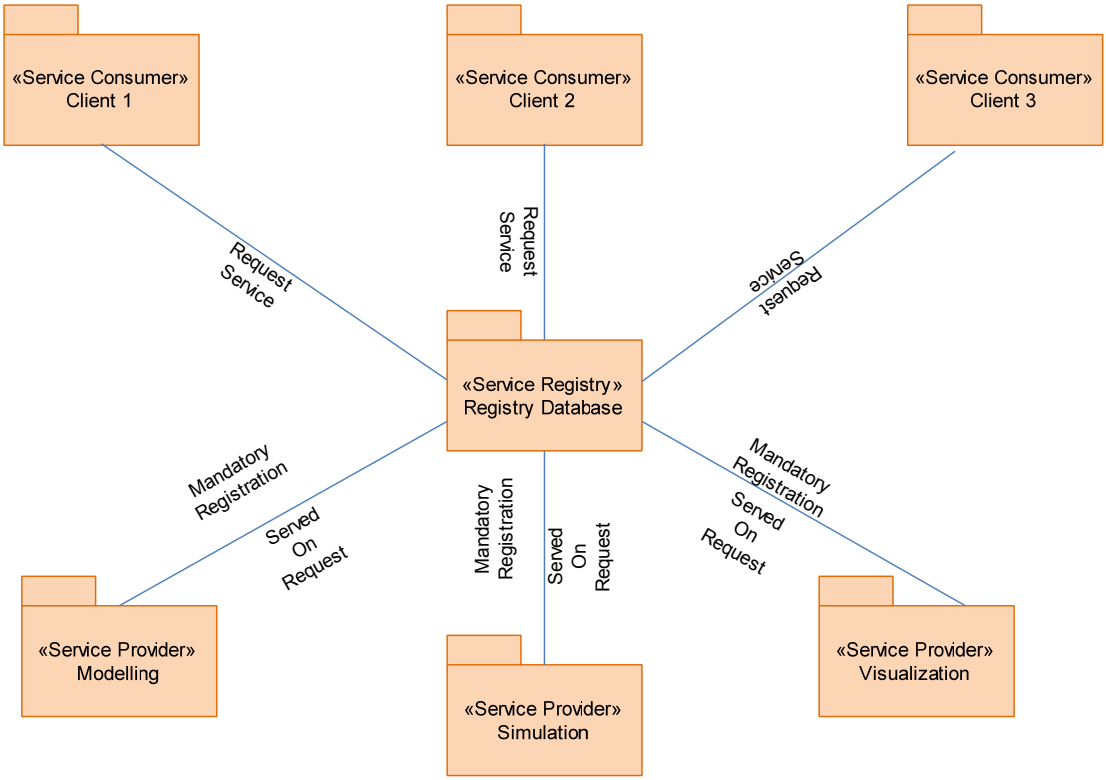

Fig. 6.5 Main packages

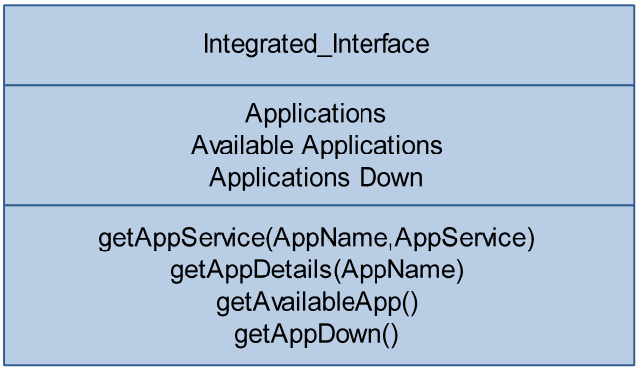

Fig. 6.6 Service Consumer class diagram

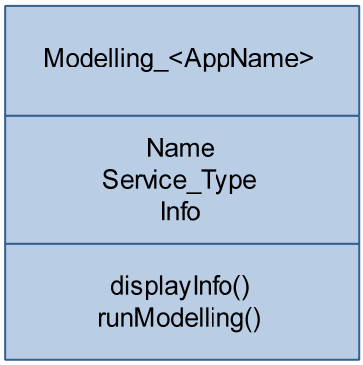

Fig. 6.7 Modelling Service Provider class diagram

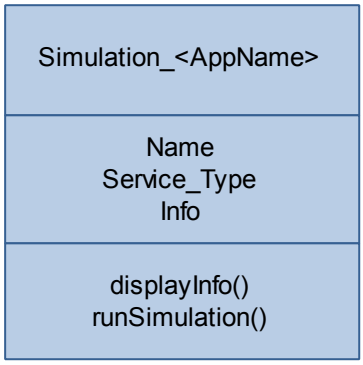

Fig. 6.8 Simulation Service Provider class diagram

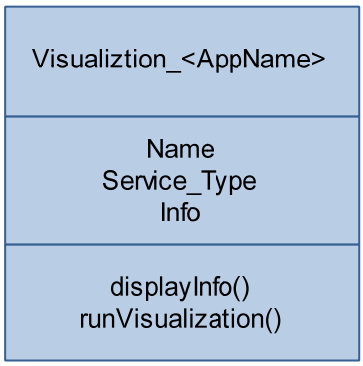

Fig. 6.9 Visualization Service Provider class diagram

6.4 Test Case Design

In this section the scenario for testing in the prototype is described as follows:

Several web services are up and running and already registered to service registry. The knowledge database needs to contain initial service profiles (initial cases) at the beginning of system operation. This is necessary since the case base does not have enough service profiles to exploit / learn from when the system is initially put into operation.

- Test 1
  - Case: prototype is up and running, without the autonomic, self-organizing feature. Web services are simulated to be down / unavailable sometimes. Users are requesting services. The tests should vary between sequential incoming requests from users and concurrent requests.
  - Input: incoming service request(s) from user(s).
  - Output:
    1. Provide the requested services if possible.
    2. Show error or warning message if the system can not provide the requested services.
    3. Provide report containing the requested services, the services provided to the users, response time, reliability, availability, and accuracy.

- Test 2
  - Case: prototype is up and running, with the autonomic, self-organizing feature. Web services are simulated to be down / unavailable sometimes. Users are requesting services. The tests should vary between sequential incoming requests from users (one request at a time) and concurrent requests. The autonomic feature should enable the system to provide the same service even when the initially intended service is unavailable. If a web service is unavailable, the system should be able to provide service from other provider seamlessly.

- Input: incoming service request(s) from user(s).
- Output:
  1. Provide the requested services if possible.
  2. Show warning message if the system can not provide the requested services.
  3. Provide report containing the requested services, the services provided to the users, response time, reliability, availability, and accuracy.

6.5 Discussion and Analysis

The following visualization applications have been successfully implemented as web services in the prototype:

- Mortality rate survey visualizer

  This is an application for user to key in survey data of mortality rate caused by different types of diseases and display the charts of those data.

  Chart example as output of the application is shown in Figure 6.10.

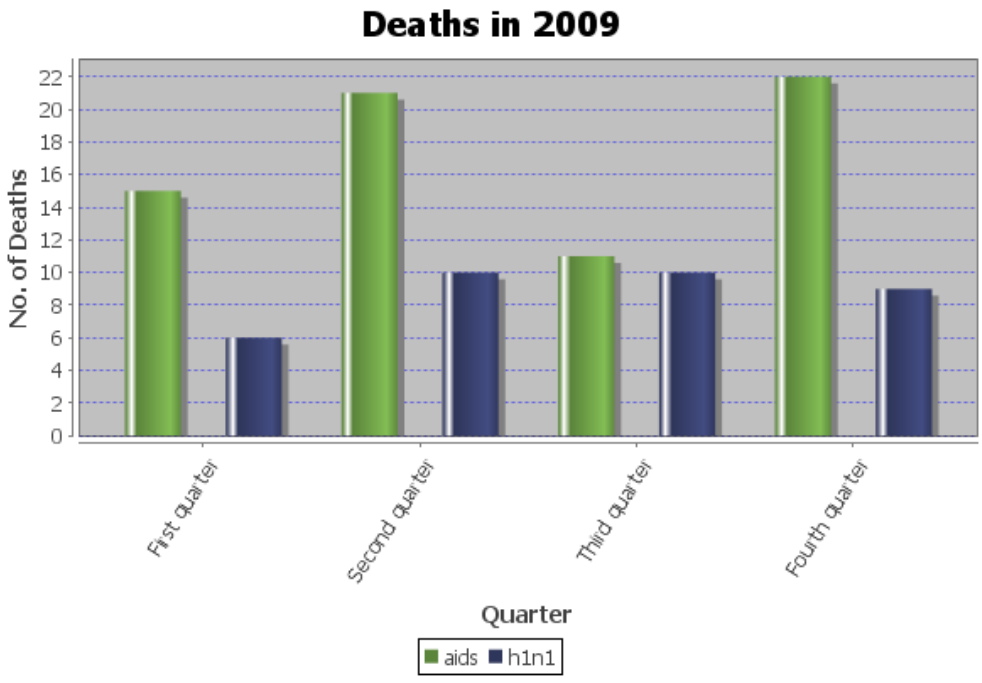

Fig. 6.10 Sample output of the survey visualize application

- SunFlow

  Sunflow is an open source rendering system for photo-realistic image synthesis. It is written in Java and built around a flexible ray tracing core and an extensible object-oriented design.

  Given a scene (*.sc) file, SunFlow will execute rendering and display the result. Figure 6.11 shows the rendering result of *cornell_box_jensen.sc* file (please refer to Appendix B for the code of the *cornell_box_jensen.sc* file).

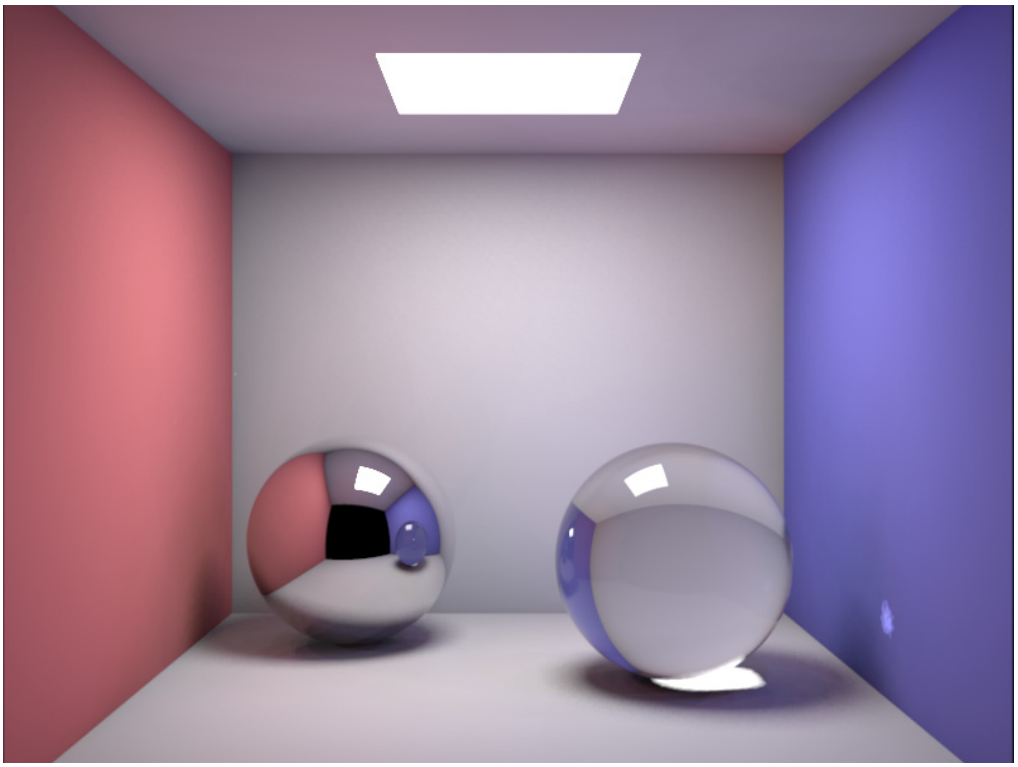

Fig. 6.11 Rendering output of *cornell_box_jensen.sc* file using SunFlow application

For the suggestion feature of the prototype, the number of usage of each application (*service_count* record) is provided in the database implementation, as well as *app_uptime* and *app_downtime* records that are to be used in measuring service availability as one of the performance metrics. The application uptime will be the difference between *app_lastdowntime* and *app_lastuptime* as the down time will be the difference between *app_lastuptime* and *app_lastdowntime*. It will follow a cycle which goes like this:

uptime = *app_currenttime* – *app_lastuptime* (if system is currently up)
or
uptime = *app_lastdowntime* – *app_lastuptime* (if system is currently down)

downtime = *app_lastuptime* – *app_lastdowntime* (if system is currently up)
or
downtime = *app_currenttime* – *app_lastdowntime* (if system is currently down)

Whenever a user attempts to find an application relevant to him or her, the user will search the knowledge base and the resulting displayed will be relevant to his or her search criteria. If it is not matching any of the applications description or keyword in the database in this prototype, then there will be no applications displayed. Please refer to the "*Knowledge_Base*" table and its columns "*description*" and "*keyword*". For example: consider the "*Image Renderer*" application. Its "*description*" and "*keyword*" are "*image renderer and visualizer*" and "*image renderer*" respectively.

If a user intends to view an image or to render and view an image or other similar cases, then if the user searches for the application relevant to his or her need, the knowledge base will be searched with the keyword entered by him or her against the "*description*" and "*keyword*". Consider user entering "*image*" then the associated application will be "*Image Renderer*".

This research was unable to obtain extensive quantitative prototype implementation results and benchmarking due to the following issues encountered during the implementation work:

- A lot more number of diverse implemented services to fully analyze the framework was required. Only two web services are implemented in the prototype and they are of the same type of services, i.e. atomic and visualisation application service.

- Non-restricted access to distributed digital service ecosystems in order to incorporate the autonomous feature into the business process layer and do thorough benchmarking were required as well.

Unfortunately until the end of this research, these requirements could not be obtained within the scope of resources of this research.

6.6 Chapter Summary

This chapter has presented the prototype design and development of the proposed autonomic SOA framework. The design is based on the UML models derived in chapter four. Then the prototype was developed using open source tools and software. These design and development works serve as proof of implementation feasibility of the proposed framework.

Unfortunately, during prototype design and development, several obstacles limited the prototype implementation, including the lacking numbers and diversity of services available to implement the proposed framework, making quantitative analysis was not possible to be done. Through the design and development experienced and described in this chapter, in spite of the limitation, it is believed that the proposed framework is feasible to be implemented in real world applications.

CHAPTER 7

CONCLUSION AND RECOMMENDATIONS

7.0 Chapter Overview

This final chapter is organized into two sections. The first section provides conclusion of the knowledge and results gained throughout this research. The second section provides some recommendations for future research direction.

7.1 Conclusion

This thesis presents a research on foundation for autonomous service-oriented architecture and is aimed to highlight the use of autonomic computing paradigm and case-based reasoning in service-oriented architecture and to develop autonomous service-oriented architecture to cope with the increased scale and complexity of future distributed systems in which SOA is a part of. A review on existing service-oriented architecture and the technologies involved initially is presented. It is then followed by the review of the adaptation of biological and nature inspired approach in complex and distributed computer systems.

Proposing an autonomous service-oriented framework based on autonomic computing architectural considerations is the next step in this research, in which the proposed framework combined autonomic computing cycle with case-based reasoning cycle to provide SOA with learning and adaptability features towards intelligent and autonomic SOA. The framework also incorporated mechanism to determine status of web services in the service ecosystem based on snapshot algorithm, and collaboration

mechanism between service ecosystems based on collaboration agreements. This framework design and development satisfied the first objective of this research, i.e. to design and develop an autonomic SOA framework based on the concepts of adapting self-organization / self-configuration of autonomic computing into SOA.

The models of the proposed framework were developed using UML meta-modelling and Petri Nets modelling frameworks. The Petri Nets models, serving as verification method, were then analyzed. The Petri Nets analysis showed that the proposed framework potentially able to reach $M_{success}$ (success end marking) even in the case where exceptional error message is received from web services. The framework will revise the composition plan and it will invoke the next available web service instead. This result showed that even if the external web services is not responding or giving exceptional messages, the service composition in autonomic SOA is likely will still be successful.

In this research, case studies in which the proposed framework can be applied were also provided. The case studies as validation method showed the feasibility and effectiveness of the proposed framework. It was shown that SOA can be extended and improved to cope with dynamic environment and unpredictable events that could cause services unavailability, such as crashes or network problems, by incorporating autonomic computing paradigm to monitor and analyze events and service requests, then to plan and execute the appropriate actions using the knowledge stored in database (knowledge base).

The developed UML models were then used to develop prototype of the proposed framework using open source software tools. During prototype development of the proposed framework, several obstacles that limited this research were found, including the lacking number and diversity of services available to implement and test the framework. Furthermore the SOA domain specific knowledge database that is required for the case base in the framework of the research was not yet available, and every SOA application domain would also require its own application domain specific knowledge. Even though the features of the case in the design made for this research have been described, the sufficient services and cases to build complete case base,

especially for composite type of services, were unavailable. Therefore during prototype and simulation testing, only a small number of atomic services were considered.

The simulation results showed the ability of the framework to work around unavailable atomic services and seamlessly provide user with the same type of atomic service from different service provider. If all the required atomic services, i.e. "ingredients", to compose a composite service are obtainable, then the service composition will be successful since the service composition process itself executed internally within the business process layer of the framework. Thus it can be concluded that the proposed framework will also improve the success rate of providing a composite service by ensuring the availability and reliability of its ingredients (atomic services).

With respective to the second objective of this research, i.e. to develop models and simulation / prototype of the proposed autonomous service-oriented architecture, and then analyze the simulation / prototype results, these works partially satisfied that objective. Petri Nets based and UML based models of the proposed framework have been developed and formally analyzed (Petri Nets based analysis), simulation of simple case study has been developed, and basic prototype also has been developed. However this work was unable to provide thorough quantitative results due to the limitations mentioned earlier. Nevertheless, it is believed that the overall objective of the research, i.e. to extend the capability and intelligence of SOA by adapting autonomic computing into SOA, has been achieved.

7.2 Recommendations for Future Works

There are a number of challenges that still need to be addressed in future researches in this field. The followings have been identified to be considered for future research direction:

The proposed autonomic SOA framework is yet to be implemented in real world system applications. In this research, the proposed framework has been simulated and a basic prototype has been developed. Yet to comparing it with other SOA implementation equally, it needs to be implemented in real applications. Future works could focus on implementing the proposed framework in a specific application domain, then analyzing and benchmarking it quantitatively with other SOA implementations.

The presented research analysis on the proposed framework also has not included a thorough quantitative evaluation and analysis to measure the quantitative improvements over conventional SOA framework, especially in term of Quality of Services (QoS). This is due to the obstacles and limitations mentioned earlier. The metrics have been identified and described in this thesis, and later on further quantitative study is needed after the proposed framework has been fully implemented.

The service description in the case base can be researched and extended to benefit from the Web Ontology Language, i.e. OWL, especially OWL-S. One of the aims of this semantic web service initiative is also to enable automation of web services by creating language and ontological infrastructure to support incorporation of machine understandable semantics into web services. The adaptation of web service ontology may yield a more accurate service discovery. The efforts toward the use of semantics and ontologies have been started, for example is the research by [Maximilien & Singh, 2004] that proposed a multi-agents approach which support considerations of semantics.

The proposed framework presented in this research has not taken security aspect into consideration. Research on security aspect of the proposed autonomic SOA therefore is encouraged. The main premise of SOA is to reduce or even remove application boundaries and technology differences. As applications are opened up, owned and operated by different organizations, combining these services securely, protecting the SOA infrastructure against attack [O'Neill, 2009], and risk management in SOA [Peterson, 2008] would become important issues then. WS-Security (Web Service Security) [OASIS WSS, 2006] is the main SOA security

specifications standard which uses XML Signature and XML Encryption. Other approaches in SOA security include XML Gateways which provide security for SOA by providing security processing on the network using dedicated hardware, and security risk management driven approach which emphasises on considerations raised by authentication, authorization, auditing, and assurance.

With the fast growth of cloud computing and its researches, SOA is now finding links with cloud computing, therefore the proposed SOA framework might be possible to be researched further into cloud computing domain. In many ways, the services offered by cloud computing providers are like a global SOA. One of the layers in cloud computing is the application layer called as Software as a Service (SaaS) - software that is deployed over the internet and/or to run behind a firewall on a local area network or personal computer. It delivers software application as a service over the internet. With SaaS, a provider licenses an application to customers as a service on demand. SaaS can take advantage of the proposed autonomic SOA to let software applications communicate with each other autonomously. Each software service can act either as a service provider, exposing its functionality to other applications via public brokers, or as a service requester, incorporating data and functionality from other services.

REFERENCES

[Aamodt & Plaza, 1994] A. Aamodt and E. Plaza, "Case-based reasoning: foundational issues, methodological variations, and system approaches," in *AI Communications*, 7:39-59, 1994.

[Anceaume et al., 2005] E. Anceaume, X. Defago, M. Gradinariu, and M. Roy, "Towards a theory of self-organization", in *Proceedings of OPODIS 2005*, pp. 146-156, 2005.

[Apache-Axis] Apache Web Services – Axis [Online]. Available: http://ws.apache.org/axis/

[Arbab, 2008] F. Arbab, "Challenges in service-oriented computing," presented at Universiti Teknologi PETRONAS, Malaysia, Sep. 2008.

[Arora et al, 2006] H. Arora, T. S. Raghu, A. Vinze, and P. Brittenham, "Collaborative Self-Configuration and Learning in Autonomic Computing Systems: Applications to Supply Chain," in *Proc. IEEE International Conference on Autonomic Computing*, June 2006.

[Arsanjani, 2005] Ali Arsanjani, "How to identify, specify, and realize services for your SOA," [Online]. Available: http://www.webservices.org/categories/enterprise/strategy_architecture/how_to_identify_specify_and_realize_services_for_your_soa/, February 2005.

[Balfagih & Hassan, 2010] Z. Balfagih and M.F.B. Hassan, "Agent based Monitoring Framework for SOA Applications Quality", in *Proceedings of International Symposium in Information Technology (ITSim)*, Malaysia, 2010.

[Banzhaf, 2002] W. Banzhaf, "Self-organizing systems," *Encyclopedia of Physical Science & Technology*, Academic Press, New York, vol. 14, 2002.

[Barbera et al., 2004] M. Barbera, C. Barbero, P. D. Zovo, F. Farinaccio, E. Gkroustiotis, S. Kyriazakos, I. Mura, and G. Previti, "An Application of Case-Based Reasoning to the Adaptive Management of Wireless Networks", in *Proceedings of the 6th European Conference on Case-Based Reasoning (ECCBR), Lecture Notes in Artifical Intelligence (LNAI) 2416*, pp. 490–504, Springer-Verlag Berlin Heidelberg, 2002.

[Bass et al, 2001] L. Bass, B. E. John, and J. Kates, "Achieving Usability Through Software Architecture," Technical Report CMU/SEI-2001-TR-005 ESC-TR-2001-005, March 2001.

[Bell, 2008] M. Bell, *Service-Oriented Modeling: Service Analysis, Design, and Architecture*, John Wiley & Sons, Inc., Hoboken, New Jersey, 2008.

[Bergman, 2000] R. Bergman, "Introduction to Case-Based Reasoning", Centre for Learning Systems and Applications, University of Kaiserslautern, Germany, 2000.

[Booth et al., 2004] D. Booth, H. Haas, and A. Brown, "Web Services Glossary," Technical Report, World Wide Web Consortium (W3C) (2004) [Online]. Available: www.w3.org/TR/ws-gloss/

[Champrasert & Suzuki, 2005] P. Champrasert and J. Suzuki, "Making Grid System Self-Organizing and Adaptive: An Aprroach Leveraging Biological Concepts and Mechanisms", in *Proceedings of the 4th IASTED International Conference On Communications, Internet, and IT (CIIT),* Cambridge, Massachusetts, USA, 2005.

[Chandy & Lamport, 1985] K. M. Chandy and L. Lamport, "Distributed Snapshots: Determining Global States of Distributed Systems," *ACM Transaction on Computer Systems*, vol. 3, no. 1, pp. 63-75, February 1985.

[Cheetham, 2004] W. Cheetham, "Tenth Anniversary of the Plastics Color Formulation Tool", in *Proceedings of the 16th Innovative Applications of Artificial Intelligence Conference*, Published by The AAAI Press, California, July 2004.

[Chinnici et al., 2007] R. Chinnici, J. Moreau, A. Ryman, and S. Weerawarana (Editors), "Web services description language (WSDL) version 2.0," W3C Recommendation 26 Jun. 2007.

[Christensen et al., 2001] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web services description language (WSDL) 1.1," W3C Note 15 Mar. 2001.

[CPNTools] CPN Tools – Computer Tool for Coloured Petri Nets, [Online]. Available: http://wiki.daimi.au.dk/cpntools/cpntools.wiki

[Currency Convertor] Currency Convertor web service [Online]. Available: http://www.webservicex.net/CurrencyConvertor.asmx?wsdl

[Currency Service] Currency Service web service [Online]. Available: http://www.restfulwebservices.net/wcf/CurrencyService.svc?wsdl

[Daskalakis & Mantas, 2009] S. Daskalakis and J. Mantas, "The Impact of SOA for Achieving Healthcare Interoperability", *Methods of Information in Medicine*, Vol. 48, Issue 2, pp. 190-195, Schattauer Publishers, Stuttgart, 2009.

[Erl, 2007] T. Erl, SOA Principles of Service Design, Prentice Hall / Pearson PTR, ISBN: 0132344823, 1st Edition July 2007.

[Everware-CBDI, 2011] Everware-CBDI, "CBD-Service Architecture and Engineering (SAE) Meta Model for SOA version 3.0", Feb. 2011.

[Fielding, 2000] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures", Doctoral Dissertation, University of California, Irvine, 2000.

[Georgiadis et al., 2002] I. Georgiadis, J. Magee, and J. Kramer, "Self-Organizing Software Architectures for Distributed Systems", in *Proceedings of ACM SIGSOFT 2002 Workshop on Self-Healing Systems* (WOSS 2002), Charleston, USA, 2002.

[Goudar, 2008] A. Goudar, "SOA Measurements and Reporting," White Paper, MPHASIS, December 2008.

[Gurguis & Zeid, 2005] S. A. Gurguis and A. Zeid, "Towards Autonomic Web Services: Achieving Self-Healing Using Web Services", in *Design and Evolution of Autonomic Application Software (DEAS 2005)*, St. Louis, Missouri, USA, 2005.

[Haas & Brown, 2004] H. Haas and A. Brown (Editors), "Web services glossary," W3C Working Group Note 11 Feb. 2004.

[Hart et al., 2007] E. Hart, D. Davoudani, and C. McEwan, “Immunological inspiration for building a new generation of autonomic systems,” in *Proceedings of the 1st International Conference of Autonomic Computing & Communication Systems*, Rome, Italy, 2007.

[He, 2003] H. He, “What Is Service-Oriented Architecture,” XML.com, O'Reilly Media, Inc., 2003 [Online]. Available: http://www.xml.com/pub/a/ws/2003/09/30/soa.html

[Heydarnoori et al., 2006] A. Heydarnoori, F. Mavaddat, and F. Arbab, “Towards an automated deployment planner for composition of web services as software components,” *Electronic Notes in Theoretical Computer Sciences*, Elsevier B.V., 2006.

[Hinkle & Toomey, 1995] D. Hinkle and C. Toomey, “Applying Case-Based Reasoning to Manufacturing”, *AI Magazine* 16(1), pp. 65-73, Spring, 1995.

[Hu, 2006] S. X. K. Hu, “Interoperability at the SOAP message level: A WSDL Design Case Study”, 2006 [Online]. Available: http://www.ibm.com/developerworks/webservices/library/ws-soa-intersoap/

[Huhns & Singh, 2005] M. N. Huhns and M. P. Singh, “Service-oriented computing: key concepts and principles,” in *IEEE Internet Computing*, Jan-Feb. 2005, pp. 75-81.

[IBM, 2001] IBM, “Autonomic computing: IBM’s perspective on the state of information technology,” USA, October 2001.

[Juneja et al., 2008] G. Juneja, B. Dournaee, J. Natoli, and S. Birkel, “SOA in Healthcare (Part I)”, SOA Magazine Issue XVII, April 2008.

[Juneja et al., 2009] G. Juneja, B. Dournaee, J. Natoli, and S. Birkel, “SOA in Healthcare (Part II)”, SOA Magazine Issue XXVII, March 2009.

[Kephart & Chess, 2003] J. O Kephart and D. M. Chess, “The vision of autonomic computing,” in *Computer*, vol. 36, No. 1, IEEE Computer Society, pp. 41-50, Jan. 2003.

[Kim et al., 2006] H. S. Kim, S. H. Kuk, J.-K. Lee, and S.-W. Park. "Construction of a Service-Oriented Architecture based e-Engineering Framework," in *Proceedings of IASTED International Conferences: Web Technologies, Applications, and Services*, Calgary, Canada, Jul. 2006.

[Kolodner, 1992] J. L. Kolodner, "An Introduction to Case-Based Reasoning", *Artificial Intelligence Review*, vol. 6, pp. 3-34, 1992.

[Krasnogor & Gheorghe, 2005] N. Krasnogor and M. Gheorghe, "Systems self-assembly," The Grand Challenge in Non-Classical Computation International Workshop, University of York, Apr. 2005.

[Lazovik & Arbab, 2007] A. Lazovik and F. Arbab, "Using Reo for service coordination," in *Proceedings of ICSOS 2007*, LNCS 4749, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 398-403.

[Lynch, 2007] J. Lynch, "SOA – Myth or Reality", presented at AIX and Linux Technical University, San Antonio, Texas, USA, 2007.

[Maximilien & Singh, 2004] E. M. Maximilien and M. P. Singh, "Toward Autonomic Web Services Trust and Selection", in *the 2nd International Conference on Service Oriented Computing (ICSOC'04)*, New York, USA, Nov. 2008.

[Montani & Anglano, 2008] S. Montani and C. Anglano, "Achieving self-healing in service delivery software systems by means of case-based reasoning," *Journal of Applied Intelligence*, vol. 28, No. 2, Springer Netherland, Apr. 2008, pp. 139-152.

[Montresor et al., 2002] A. Montresor, H. Meling, and O. Babaoglu, "Toward self-organizing, self-repairing, and resilient large-scale distributed systems," Technical Report UBLCS-2002-10, Sep. 2002.

[Morgan et al., 2004] A. P. Morgan, J. A. Cafeo, K. Godden, R. M. Lesperance, A. M. Simon, D. L. McGuinness, and J. L. Benedict, "The General Motors Variation-Reduction Adviser: Deployment Issues for an AI Application", in *Proceedings of the 16th Innovative Applications of Artificial Intelligence Conference*, AAAI Press, California, July 2004.

[Murata, 1989] T. Murata, "Petri Nets: Properties, Analysis, and Applications," Proceedings of the IEEE, vol. 77, no. 4, April 1989.

[O'Neill, 2009] M. O'Neill, "SOA Security: The Basics", March 2009 [Online]. Available: http://www.csoonline.com/article/484120/soa-security-the-basics

[OASIS Standard, 2003] OASIS, "Universal Description, Discovery and Integration (UDDI) v2," OASIS Standard, Apr. 2003.

[OASIS Standard, 2005] OASIS, "Universal Description, Discovery and Integration (UDDI) v3.0.2," OASIS Standard, Feb. 2005.

[OASIS Standard, 2006] OASIS, "Reference Model for Service Oriented Architecture", OASIS Standard, 12 Oct. 2006.

[OASIS WSS, 2006] OASIS, "Web Services Security (WSS)", OASIS Standard, Nov. 2006.

[Parashar & Hariri, 2005] M. Parashar and S. Hariri, "Autonomic Computing: An Overview", J.-P. Banˆatre et al. (Eds.): *UPP 2004, LNCS 3566*, pp. 247–259, Springer-Verlag Berlin Heidelberg 2005

[Perepletchikov et al., 2007] M. Perepletchikov, C. Ryan, and K. Frampton, "Cohesion Metrics for Predicting Maintainability of Service-Oriented Software," in *Proceedings of the 7th International Conference on Quality Software (QSIC)*, 2007.

[Peterson, 2008] G. Peterson, "Security in SOA", SOA Magazine Issue XV, February 2008.

[Petri, 1966] C. A. Petri, "Communication with Automata," New York: Griffiss Air Force Base. Tech. Rep. RADC-TR-65-377, vol. 1, suppl. 1, 1966.

[Poggi et al., 2006] A. Poggi, M. Tomaiuolo, P. Turci, "An Agent-Based Service Oriented Architecture", in *Proceedings of WOA 2007*, pp. 157-165, Genova, Italy, 24-25 September 2007.

[Ricci & Denti, 2007] A. Ricci and E. Denti, "simpA-WS: A Simple Agent-Oriented Programming Model & Technology for Developing SOA & Web Services", in *Proceedings of WOA 2007*, pp. 140-156, Genova, Italy, 24-25 September 2007.

[Ricci et al., 2006] A. Ricci, C. Buda, N. Zaghini, A. Natali, M. Viroli, A. Omicini, "simpA-WS: An Agent-Oriented Computing Technology for WS-based SOA Applications", in *Proceedings of the 7th WOA 2006 Workshop: From Objects to Agents*, Catania, Italy, September 26-27, 2006.

[Rosen et al., 2008] M. Rosen, B. Lublinsky, K. T. Smith, and M. J. Balcer, *Applied SOA: Service-Oriented Architecture and Design Strategies*, Wiley Publishing, Inc., Indianapolis, Indiana, USA, 2008.

[Russell et al., 2006] D. J. Russell, N. Looker, J. Xu, "SOA, Dependability, and Measures and Metrics for Network Enabled Capability," NECTISE Project Report, University of Leeds, UK, 2006.

[Schank, 1982] R. C. Shank, "Dynamic memory: A theory of learning in people and computers", Cambridge: Cambridge University Press, 1982.

[Schneider et al., 2008] D. Schneider, C. Bunse, and K. Schmid, "Towards Adaptive Service Engineering", *Proceedings of the International Workshop on the Foundations of Service-Oriented Architecture*, Special Report CMU/SEI-2008-SR-011, June 2008.

[Serugendo et al., 2006] G. Di Marzo Serugendo, J. Fitzgerald, A. Romanovsky, and N. Guelfi, "Dependable self-organising software architecture – an approach for self-managing systems," Technical Report, School of Computer Science & Information Systems, Birkbeck College, London, UK, May 2006.

[Shen et al., 2007] W. Shen, Q. Hao, S. Wang, Y. Li, and H. Ghenniwa, "An agent-based service-oriented integration architecture for collaborative intelligent manufacturing", *Journal Robotics and Computer-Integrated Manufacturing*, Vol. 23, Issue 3, June, Pergamon Press, Inc. Tarrytown, NY, USA, 2007.

[Smith & Lewis, 2009] D. B. Smith and G. A. Lewis, "SOA for Healthcare, Promises and pitfalls", [Online]. Available: http://www.asianhhm.com/information_technology/soa-healthcare.htm

[SunFlow] SunFlow – Global Illumination Rendering System, [Online]. Available: http://sunflow.sourceforge.net/

[Tosi et al., 2009] D. Tosi, G. Denaro, and M. Pezzè, "Towards autonomic service-oriented applications", *Int. J. Autonomic Computing*, Vol. 1, No. 1, pp. 58–80, Inderscience Enterprises Ltd., 2009.

[Vitvar et al., 2007] T. Vitvar, A. Mocan, M. Kerrigan, M. Zaremba, M. Zaremba, M. Moran, and E. Cimpian, "Semantically-enabled service oriented architecture: concepts, technology and application", *Service Oriented Computing and Applications*, Vol. 1, No. 2, pp. 129-154, Springer, 2007.

[Wang, 2007] Y. Wang, "Toward Theoretical Foundations of Autonomic Computing", *Int'l Journal of Cognitive Informatics and Natural Intelligence*, 1(3), 1-16, IGI Global, 2007.

[Watson & Gardingen, 1999] I. Watson and D. Gardingen, "A Distributed Case-Based Reasoning Application for Engineering Sales Support", in *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, Vol. 1, pp. 600-605, Morgan Kaufmann Publishers, 1999.

[White et al., 2004] S. R. White, J. E. Hanson, I. Whalley, D. M. Chess, and J. O. Kephart, "An Architectural Approach to Autonomic Computing", *Proceedings of the International Conference on Autonomic Computing (ICAC)*, 2004.

[Woolf, 2006] B. Woolf, "WebSphere SOA and JEE in Practice," [Online]. Available: https://www.ibm.com/developerworks/mydeveloperworks/blogs/woolf/entry/composite_services?lang=en, April 2006.

[Xu & Muñoz-Avila, 2004] K. Xu and H. Muñoz-Avila, "CaBMA: Case-Based Project Management Assistant", in *Proceedings of the 16th Innovative Applications of Artificial Intelligence Conference*, Published by The AAAI Press, California, July 2004.

[Yang et al., 2004] B.-S. Yang, S. K. Jeong, Y.-M. Oh, and A. C. C. Tan, "Case-based reasoning system with Petri nets for induction motor fault diagnosis", *Expert Systems with Applications* 27, pp. 301–311, Elsevier Ltd., 2004.

[Yelmo et al, 2009] J. C. Yelmo, R. Trapero, and J. M. del Alamo, "Identity management and web services as service ecosystem drivers in converged networks," *IEEE Communications Magazine*, March 2009.

[Yoo et al., 2009] T. Yoo, B. Jeong, and H. Cho, "A Petri Nets based functional validation for services composition," *Expert Systems with Applications 37 (2010)*, pp. 3768–3776, Elsevier, 2009.

[Zhang et al., 2006] T. Zhang, S. Ying, S. Cao, and X. Zhong. "Meta-modeling service oriented architecture based on UML," in *Proceedings of IASTED International Conferences: Web Technologies, Applications, and Services*, Jul. 2006, Calgary, Canada.

[Zhanhg & Zhu] H.-X. Zhanhg and L.-Z. Zhu, “Building Dynamic Model in UML using Colored Petri Nets”, in *International Symposium on Computer Network and Multimedia Technology (CNMT 2009)*, Wuhan, Jan. 2009.

[Zurowska & Deter, 2007] K. Zurowska and R. Deters, “Overcoming failures in composite web services by analysing colored petri nets,” in *CPN'07 - Workshop and Tutorial on Practical Use of Coloured Petri Nets and CPN Tools*, Denmark, 2007.

[Zwicky et al., 2000] E. D. Zwicky, S. Cooper, and D. B. Chapman, *Building Internet Firewalls*, ISBN: 1-56592-871-7, Second edition, O'Reilly Media, June 2000.

LIST OF PUBLICATIONS

1. Muhammad Agni Catur Bhakti and Azween Abdullah, “Nature-Inspired Self-Organizing Service Oriented Architecture,” in *Proceedings of the National Postgraduate Conference on Engineering, Science, Technology (NPC 2009),* Universiti Teknologi PETRONAS, Malaysia, March 2009.
2. M. Agni Catur Bhakti and Azween Abdullah, “Nature-Inspired Self-Organizing Service Oriented Architecture: A Proposal,” in *Proceedings of the 6th International Conference on Information Technology in Asia (CITA 2009),* Sarawak, Malaysia, July 2009.
3. M. Agni Catur Bhakti and Azween Abdullah, “Towards Self-Organizing Service Oriented Architecture,” in *Proceedings of IEEE Conference on Innovative Technologies in Intelligent Systems & Industrial Applications (CITISIA 2009),* Kuala Lumpur, Malaysia, July 2009.
4. M. Agni Catur Bhakti and Azween Abdullah, “Towards a Nature Inspired, Self Organizing Service Oriented Architecture,” in *MASAUM Journal of Basic and Applied Sciences*, ISSN: 2076-0841, volume 1, Issue 3, pp. 421-425, October 2009.
5. M. Agni Catur Bhakti, Azween B. Abdullah, Low Tan Jung, "Autonomic, Self-Organizing Service Oriented Architecture in Service Ecosystem," *IEEE DEST 2010,* Dubai, United Arab Emirates, 12-15 April 2010.
6. M. Agni Catur Bhakti and Azween B. Abdullah, "Towards an Autonomic Service Oriented Architecture in Computational Engineering Framework," *10th International Conference on Information Science, Signal Processing and their Applications (ISSPA 2010),* Kuala Lumpur, Malaysia, 10-13 May 2010.

7. M. Agni Catur Bhakti and Azween B. Abdullah, "Design of an Autonomic Service Oriented Architecture," in *Proceedings of the 4th International Symposium on Information Technology (ITSim 2010),* volume 2, pp. 805-810, Kuala Lumpur, Malaysia, 15-17 June 2010.

8. M. Agni Catur Bhakti and Azween B. Abdullah, "An Autonomic Service Oriented Architecture in Computational Engineering Framework", *Journal of Advances in Computer Research*, ISSN: 2008-6148, 2 (2010), pp. 1-7, Islamic Azad University, Sari Branch – Iran, 2010.

9. M. Agni Catur Bhakti and Azween B. Abdullah, "Autonomic Computing Approach in Service Oriented Architecture", in *Proceedings of IEEE Symposium on Computers and Informatics (ISCI 2011)*, pp. 231-236, Kuala Lumpur, Malaysia, 20-22 March 2011.

10. M. Agni Catur Bhakti and Azween B. Abdullah, "Formal Modelling of an Autonomic Service Oriented Architecture," in *Proceedings of International Conference on Telecommunication Technology and Applications (ICTTA 2011),* Sydney, Australia, 2-3 May 2011.

APPENDIX A

SCREENSHOTS

In this section we provided some screenshots of the simulation and prototype interfaces.

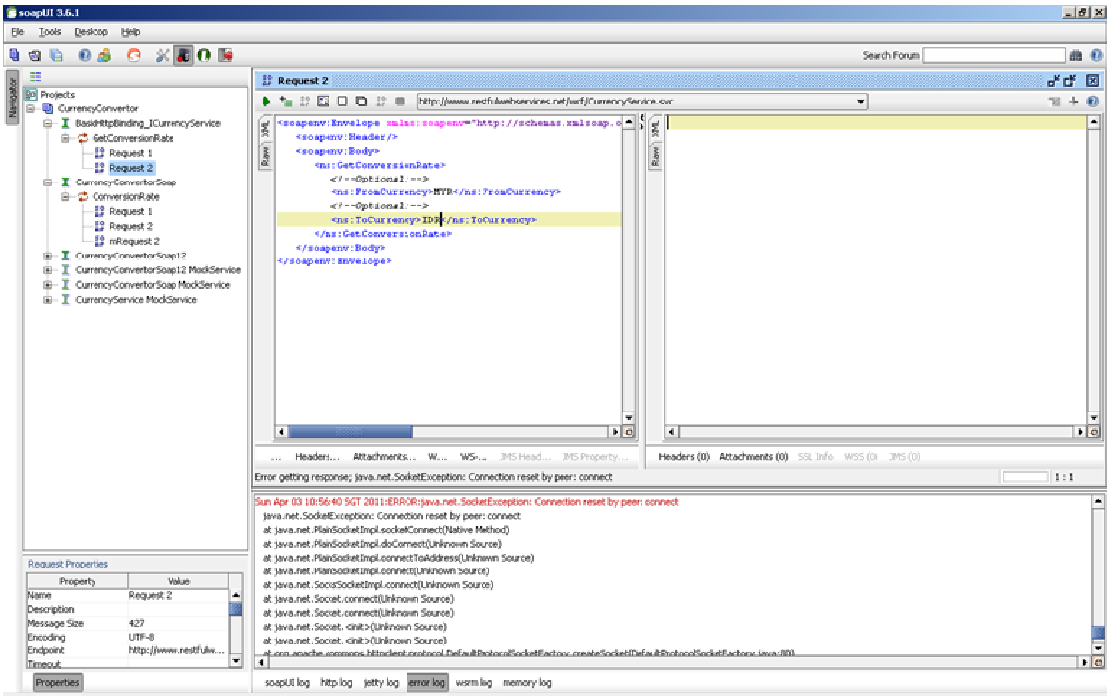

Fig. A.1 Simulation screenshot 1

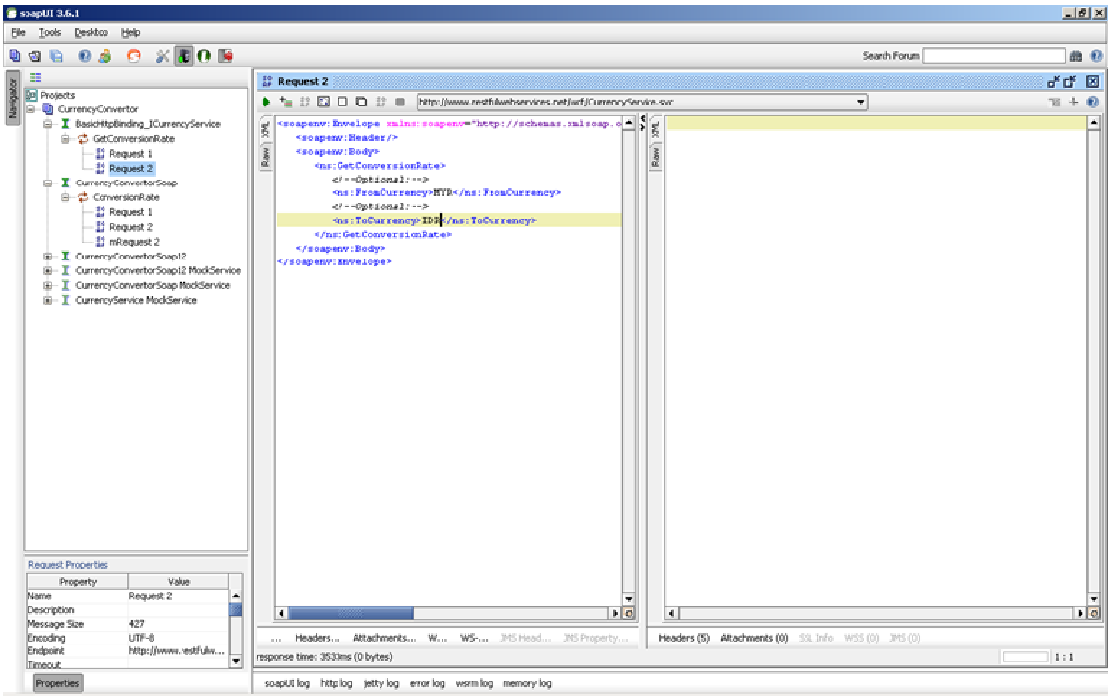

Fig. A.2 Simulation screenshot 2

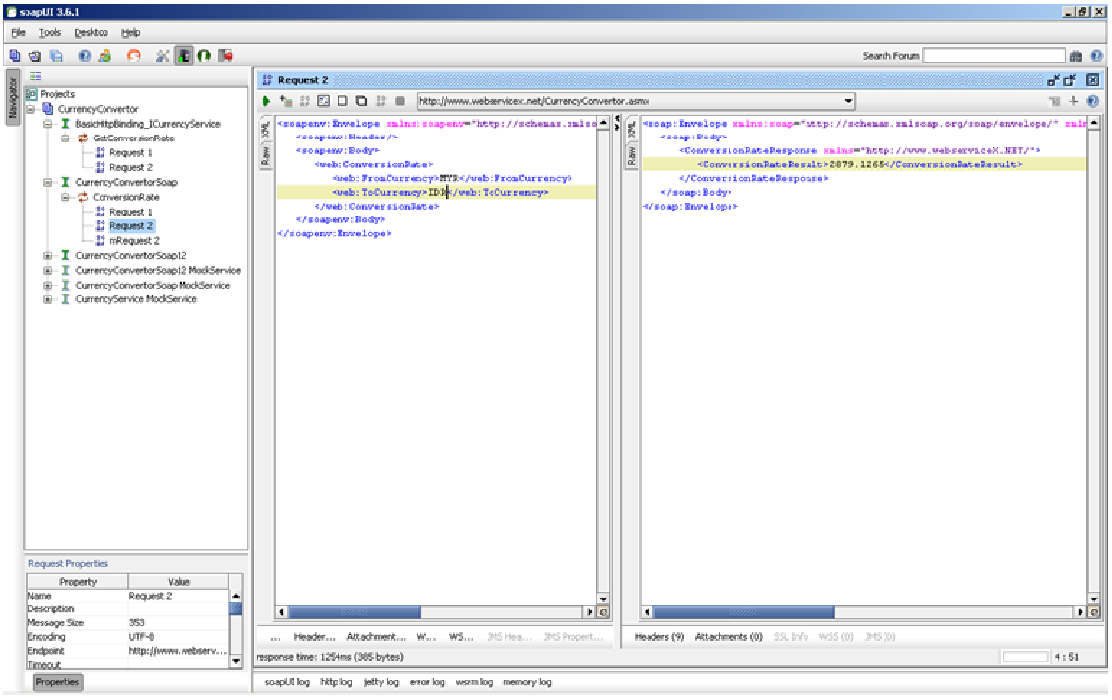

Fig. A.3 Simulation screenshot 3

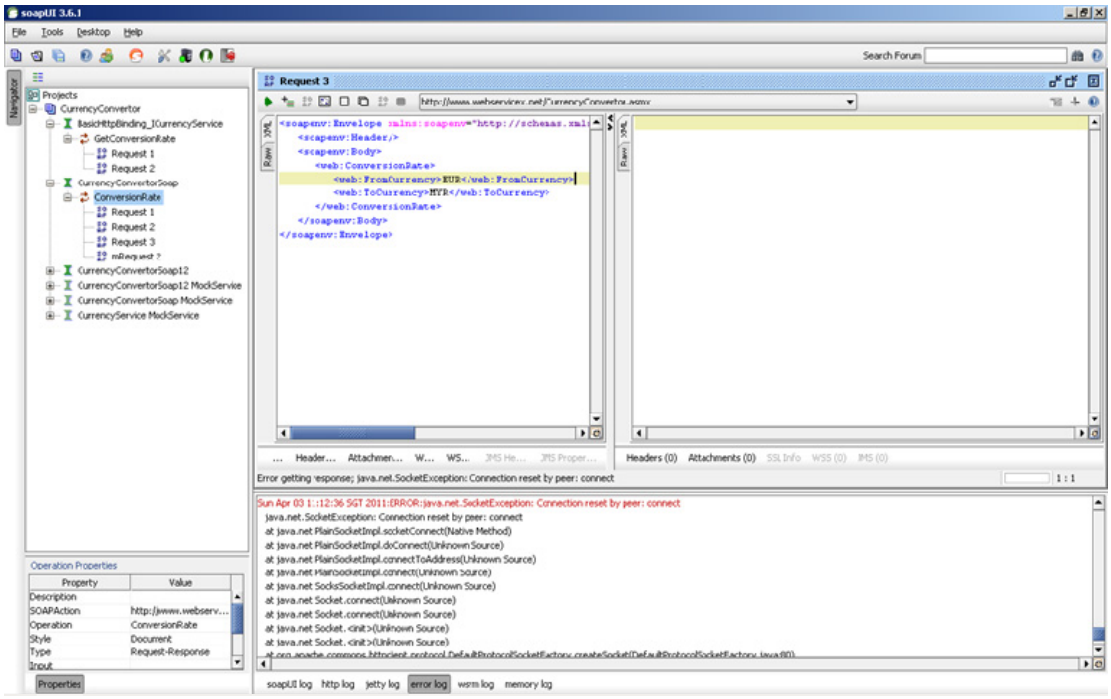

Fig. A.4 Simulation screenshot 4

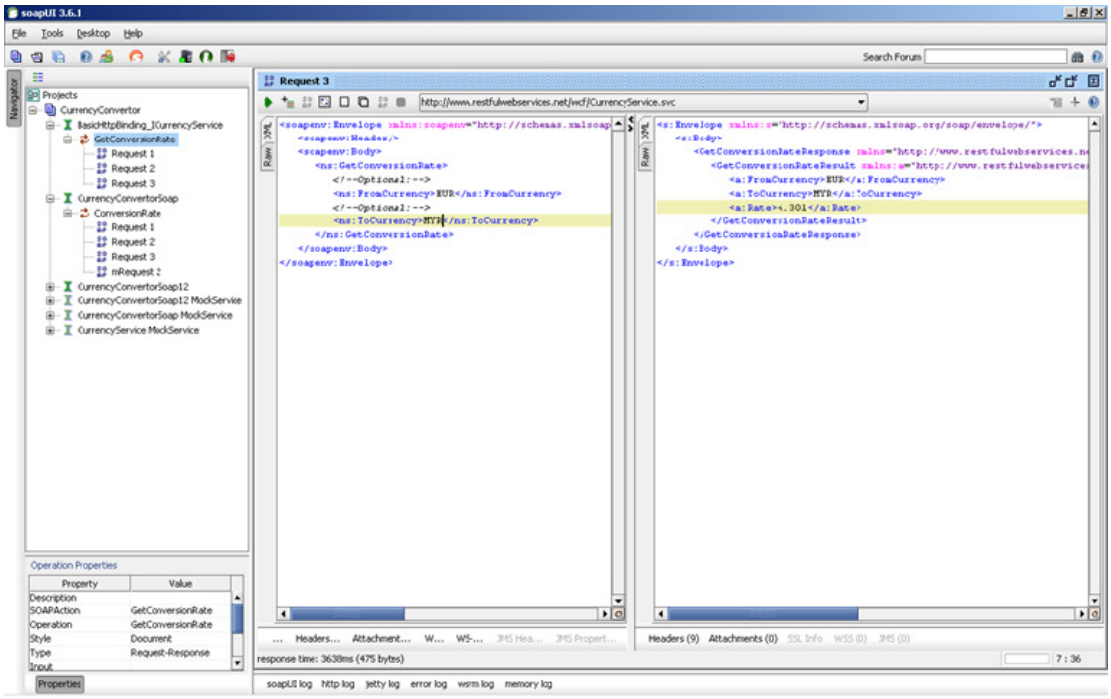

Fig. A.5 Simulation screenshot 5

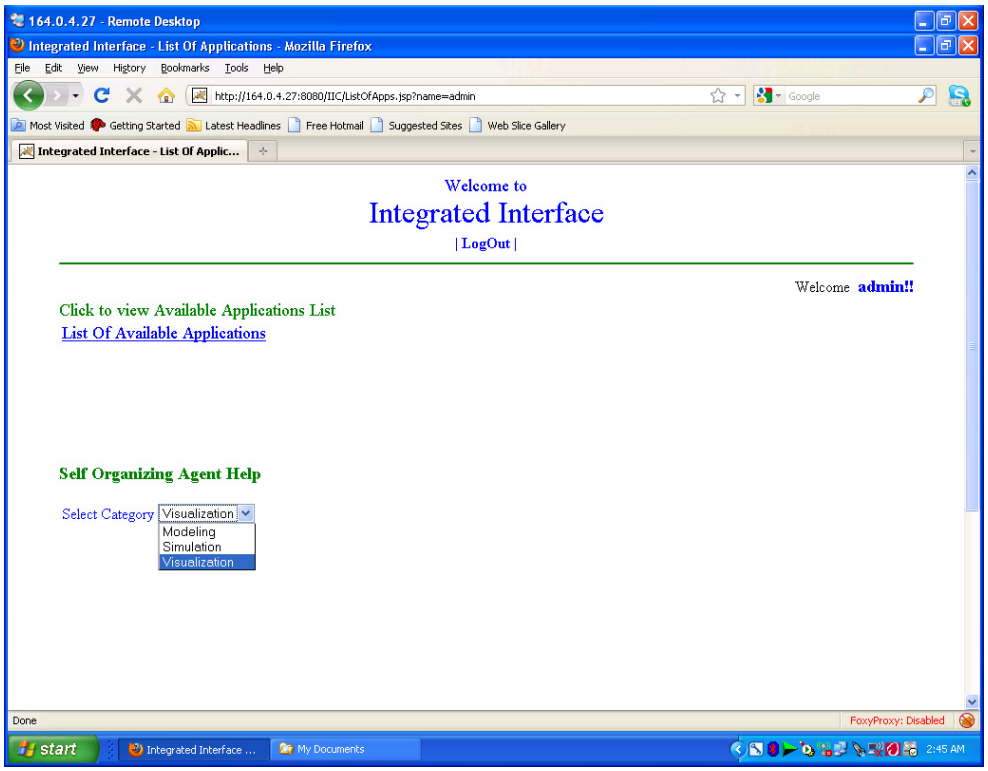

Fig. A.6 Prototype screenshot 1

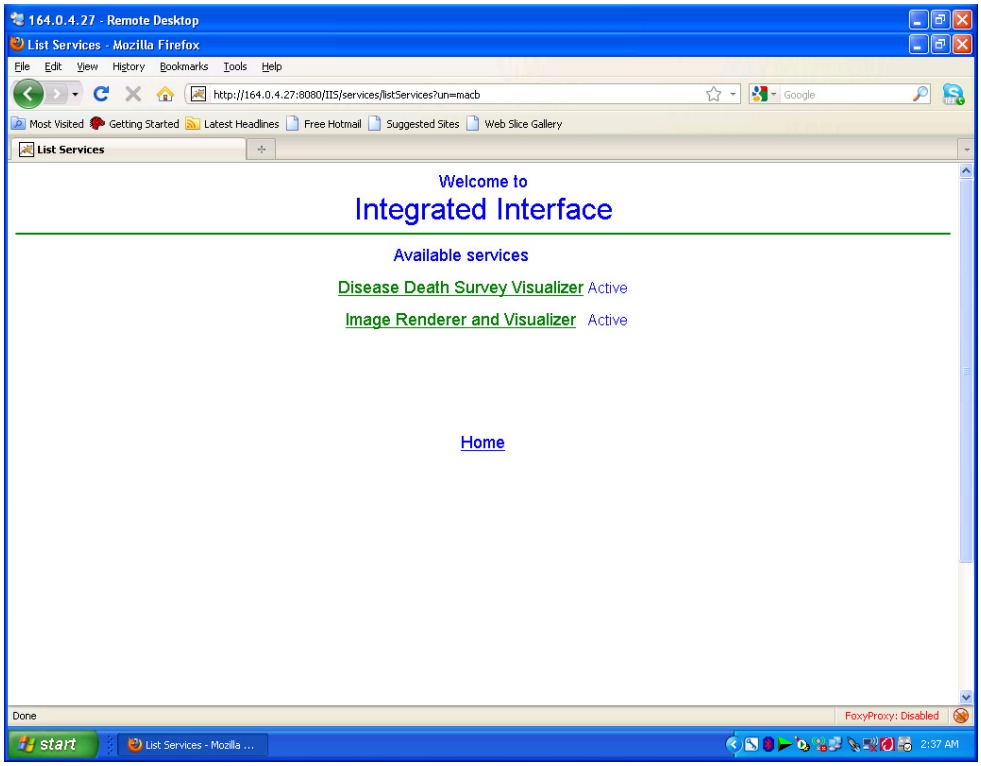

Fig. A.7 Prototype screenshot 2

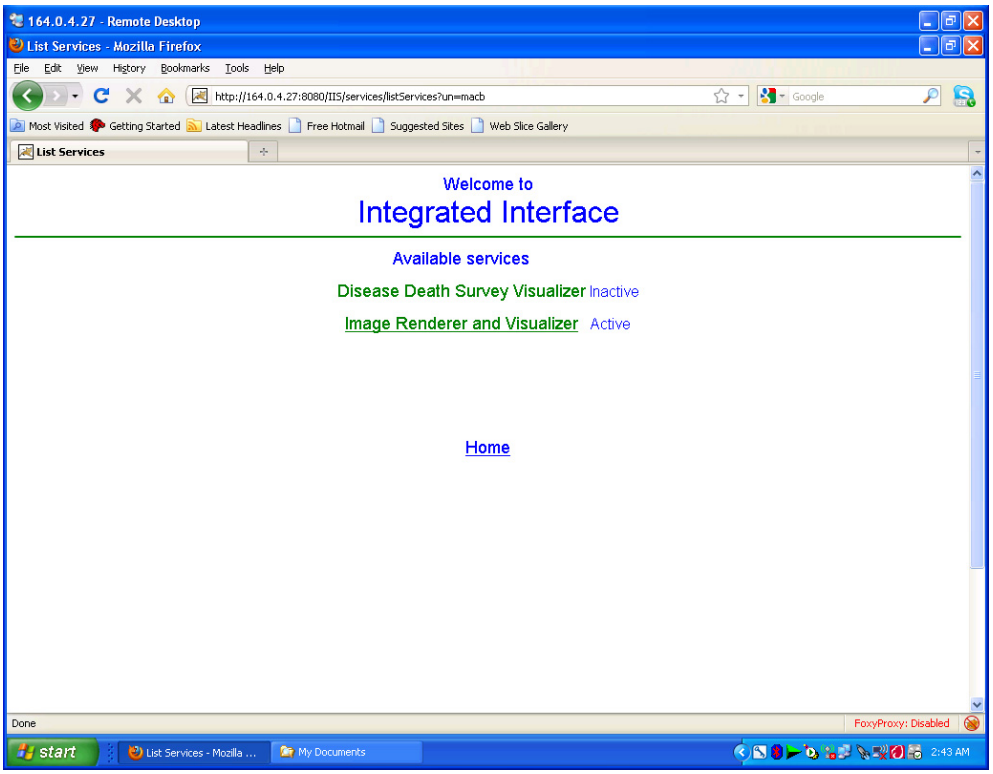

Fig. A.8 Prototype screenshot 3

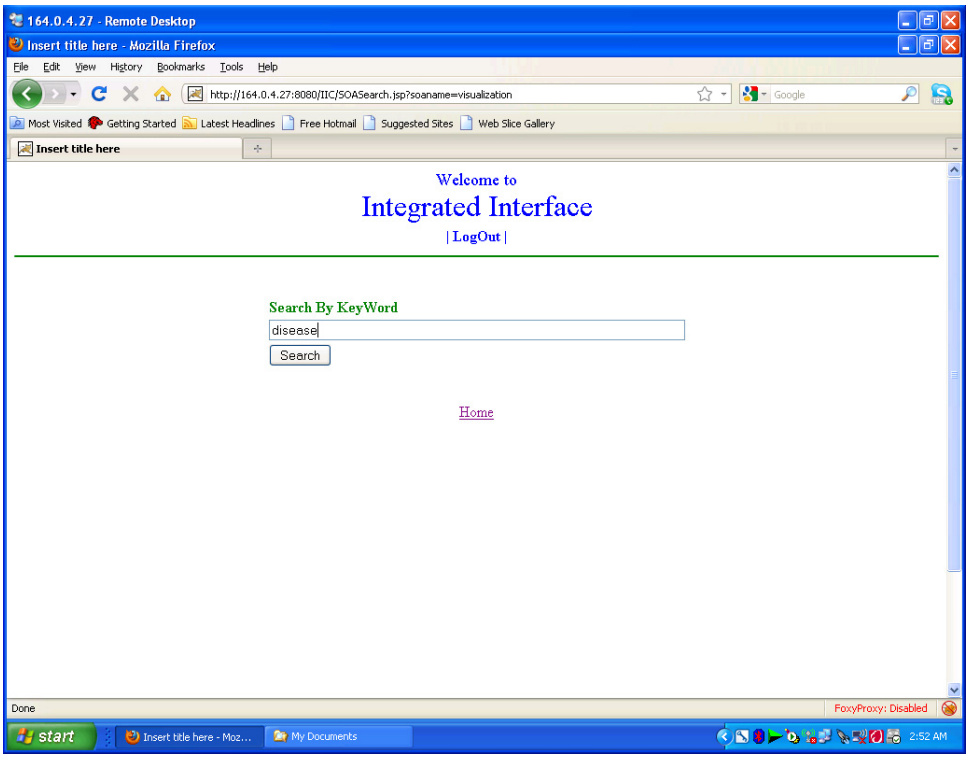

Fig. A.9 Prototype screenshot 4

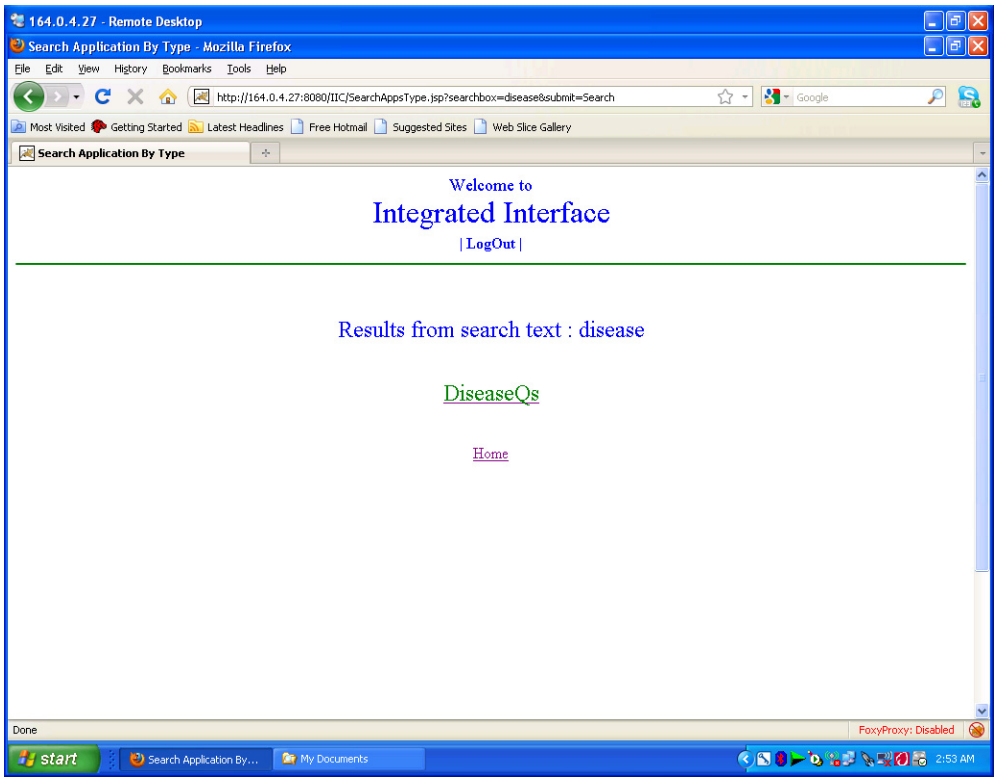

Fig. A.10 Prototype screenshot 5

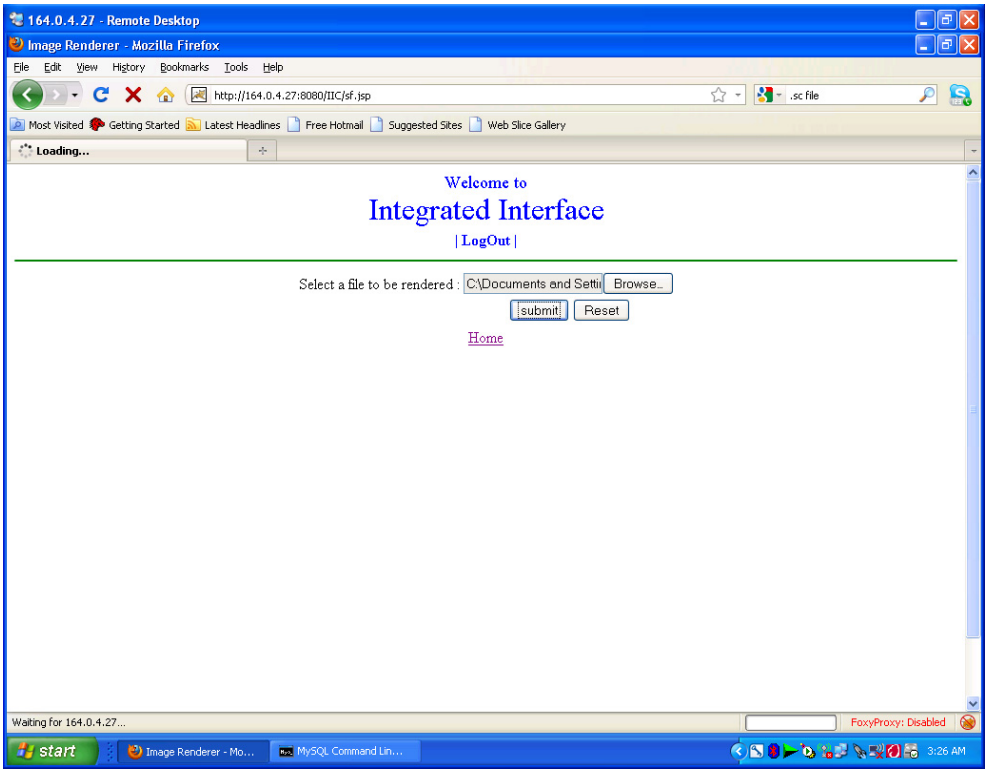

Fig. A.11 Prototype screenshot 6

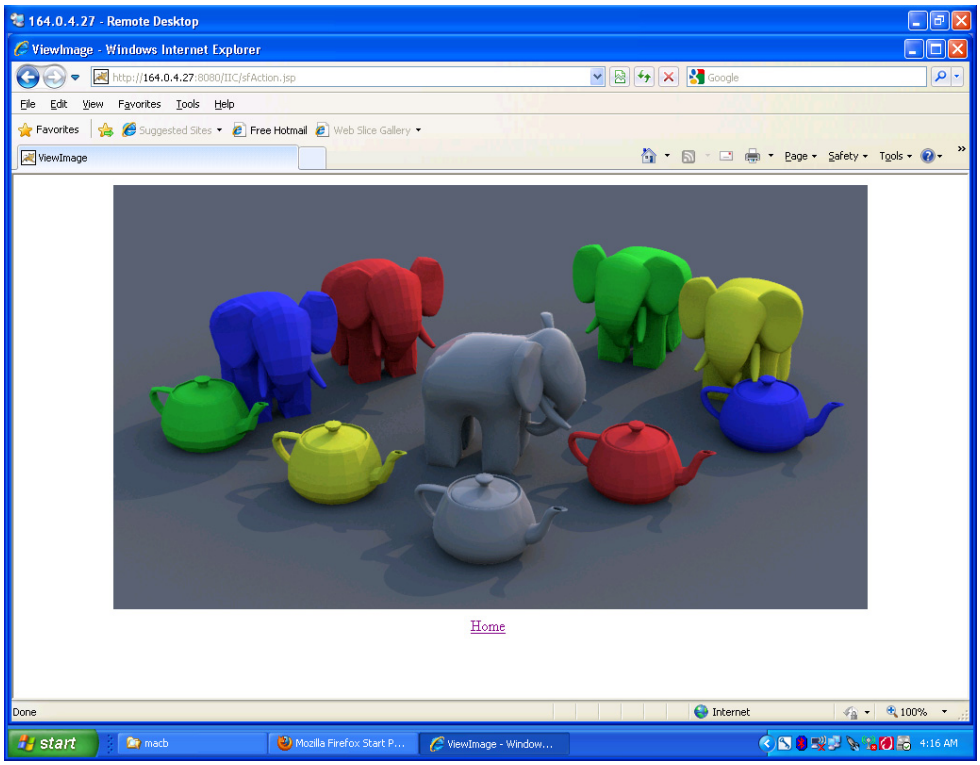

Fig. A.12 Prototype screenshot 7

APPENDIX B

SOURCE CODE

In this section we provided several main source codes of the simulation and prototype programs.

- WSDL code of *CurrencyConvertor*:

<wsdl:definitions targetNamespace="http://www.webserviceX.NET/"><wsdl:types><s:schema elementFormDefault="qualified" targetNamespace="http://www.webserviceX.NET/"><s:element name="ConversionRate"><s:complexType><s:sequence><s:element minOccurs="1" maxOccurs="1" name="FromCurrency" type="tns:Currency"/><s:element minOccurs="1" maxOccurs="1" name="ToCurrency" type="tns:Currency"/></s:sequence></s:complexType></s:element><s:simpleType name="Currency"><s:restriction base="s:string"><s:enumeration value="AFA"/><s:enumeration value="ALL"/><s:enumeration value="DZD"/><s:enumeration value="ARS"/><s:enumeration value="AWG"/><s:enumeration value="AUD"/><s:enumeration value="BSD"/><s:enumeration value="BHD"/><s:enumeration value="BDT"/><s:enumeration value="BBD"/><s:enumeration value="BZD"/><s:enumeration value="BMD"/><s:enumeration value="BTN"/><s:enumeration value="BOB"/><s:enumeration value="BWP"/><s:enumeration value="BRL"/><s:enumeration value="GBP"/><s:enumeration value="BND"/><s:enumeration value="BIF"/><s:enumeration value="XOF"/><s:enumeration value="XAF"/><s:enumeration value="KHR"/><s:enumeration value="CAD"/><s:enumeration value="CVE"/><s:enumeration value="KYD"/><s:enumeration value="CLP"/><s:enumeration value="CNY"/><s:enumeration value="COP"/><s:enumeration value="KMF"/><s:enumeration value="CRC"/><s:enumeration

value="HRK"/><s:enumeration value="CUP"/><s:enumeration
value="CYP"/><s:enumeration value="CZK"/><s:enumeration
value="DKK"/><s:enumeration value="DJF"/><s:enumeration
value="DOP"/><s:enumeration value="XCD"/><s:enumeration
value="EGP"/><s:enumeration value="SVC"/><s:enumeration
value="EEK"/><s:enumeration value="ETB"/><s:enumeration
value="EUR"/><s:enumeration value="FKP"/><s:enumeration
value="GMD"/><s:enumeration value="GHC"/><s:enumeration
value="GIP"/><s:enumeration value="XAU"/><s:enumeration
value="GTQ"/><s:enumeration value="GNF"/><s:enumeration
value="GYD"/><s:enumeration value="HTG"/><s:enumeration
value="HNL"/><s:enumeration value="HKD"/><s:enumeration
value="HUF"/><s:enumeration value="ISK"/><s:enumeration
value="INR"/><s:enumeration value="IDR"/><s:enumeration
value="IQD"/><s:enumeration value="ILS"/><s:enumeration
value="JMD"/><s:enumeration value="JPY"/><s:enumeration
value="JOD"/><s:enumeration value="KZT"/><s:enumeration
value="KES"/><s:enumeration value="KRW"/><s:enumeration
value="KWD"/><s:enumeration value="LAK"/><s:enumeration
value="LVL"/><s:enumeration value="LBP"/><s:enumeration
value="LSL"/><s:enumeration value="LRD"/><s:enumeration
value="LYD"/><s:enumeration value="LTL"/><s:enumeration
value="MOP"/><s:enumeration value="MKD"/><s:enumeration
value="MGF"/><s:enumeration value="MWK"/><s:enumeration
value="MYR"/><s:enumeration value="MVR"/><s:enumeration
value="MTL"/><s:enumeration value="MRO"/><s:enumeration
value="MUR"/><s:enumeration value="MXN"/><s:enumeration
value="MDL"/><s:enumeration value="MNT"/><s:enumeration
value="MAD"/><s:enumeration value="MZM"/><s:enumeration
value="MMK"/><s:enumeration value="NAD"/><s:enumeration
value="NPR"/><s:enumeration value="ANG"/><s:enumeration
value="NZD"/><s:enumeration value="NIO"/><s:enumeration
value="NGN"/><s:enumeration value="KPW"/><s:enumeration
value="NOK"/><s:enumeration value="OMR"/><s:enumeration
value="XPF"/><s:enumeration value="PKR"/><s:enumeration
value="XPD"/><s:enumeration value="PAB"/><s:enumeration
value="PGK"/><s:enumeration value="PYG"/><s:enumeration
value="PEN"/><s:enumeration value="PHP"/><s:enumeration
value="XPT"/><s:enumeration value="PLN"/><s:enumeration

value="QAR"/><s:enumeration value="ROL"/><s:enumeration value="RUB"/><s:enumeration value="WST"/><s:enumeration value="STD"/><s:enumeration value="SAR"/><s:enumeration value="SCR"/><s:enumeration value="SLL"/><s:enumeration value="XAG"/><s:enumeration value="SGD"/><s:enumeration value="SKK"/><s:enumeration value="SIT"/><s:enumeration value="SBD"/><s:enumeration value="SOS"/><s:enumeration value="ZAR"/><s:enumeration value="LKR"/><s:enumeration value="SHP"/><s:enumeration value="SDD"/><s:enumeration value="SRG"/><s:enumeration value="SZL"/><s:enumeration value="SEK"/><s:enumeration value="CHF"/><s:enumeration value="SYP"/><s:enumeration value="TWD"/><s:enumeration value="TZS"/><s:enumeration value="THB"/><s:enumeration value="TOP"/><s:enumeration value="TTD"/><s:enumeration value="TND"/><s:enumeration value="TRL"/><s:enumeration value="USD"/><s:enumeration value="AED"/><s:enumeration value="UGX"/><s:enumeration value="UAH"/><s:enumeration value="UYU"/><s:enumeration value="VUV"/><s:enumeration value="VEB"/><s:enumeration value="VND"/><s:enumeration value="YER"/><s:enumeration value="YUM"/><s:enumeration value="ZMK"/><s:enumeration value="ZWD"/><s:enumeration value="TRY"/></s:restriction></s:simpleType><s:element name="ConversionRateResponse"><s:complexType><s:sequence><s:element minOccurs="1" maxOccurs="1" name="ConversionRateResult" type="s:double"/></s:sequence></s:complexType></s:element><s:element name="double" type="s:double"/></s:schema></wsdl:types><wsdl:message name="ConversionRateSoapIn"><wsdl:part name="parameters" element="tns:ConversionRate"/></wsdl:message><wsdl:message name="ConversionRateSoapOut"><wsdl:part name="parameters" element="tns:ConversionRateResponse"/></wsdl:message><wsdl:message name="ConversionRateHttpGetIn"><wsdl:part name="FromCurrency" type="s:string"/><wsdl:part name="ToCurrency" type="s:string"/></wsdl:message><wsdl:message name="ConversionRateHttpGetOut"><wsdl:part name="Body" element="tns:double"/></wsdl:message><wsdl:message name="ConversionRateHttpPostIn"><wsdl:part name="FromCurrency" type="s:string"/><wsdl:part name="ToCurrency" type="s:string"/></wsdl:message><wsdl:message name="ConversionRateHttpPostOut"><wsdl:part name="Body"

*element="tns:double"/></wsdl:message><wsdl:portType name="CurrencyConvertorSoap"><wsdl:operation name="ConversionRate"><wsdl:documentation>
Get conversion rate from one currency to another currency
 <u>Differenct currency Code and Names around the world</u> <blockquote> <font face='Verdana' size='1'>AFA-Afghanistan Afghani
ALL-Albanian Lek
DZD-Algerian Dinar
ARS-Argentine Peso
AWG-Aruba Florin
AUD-Australian Dollar
BSD-Bahamian Dollar
BHD-Bahraini Dinar
BDT-Bangladesh Taka
BBD-Barbados Dollar
BZD-Belize Dollar
BMD-Bermuda Dollar
BTN-Bhutan Ngultrum
BOB-Bolivian Boliviano
BWP-Botswana Pula
BRL-Brazilian Real
GBP-British Pound
BND-Brunei Dollar
BIF-Burundi Franc
XOF-CFA Franc (BCEAO)
XAF-CFA Franc (BEAC)
KHR-Cambodia Riel
CAD-Canadian Dollar
CVE-Cape Verde Escudo
KYD-Cayman Islands Dollar
CLP-Chilean Peso
CNY-Chinese Yuan
COP-Colombian Peso
KMF-Comoros Franc
CRC-Costa Rica Colon
HRK-Croatian Kuna
CUP-Cuban Peso
CYP-Cyprus Pound
CZK-Czech Koruna
DKK-Danish Krone
DJF-Dijibouti Franc
DOP-Dominican Peso
XCD-East Caribbean Dollar
EGP-Egyptian Pound
SVC-El Salvador Colon
EEK-Estonian Kroon
ETB-Ethiopian Birr
EUR-Euro
FKP-Falkland Islands Pound
GMD-Gambian Dalasi
GHC-Ghanian Cedi
GIP-Gibraltar Pound
XAU-Gold Ounces
GTQ-Guatemala Quetzal
GNF-Guinea Franc
GYD-Guyana Dollar
HTG-Haiti Gourde
HNL-Honduras Lempira
HKD-Hong Kong Dollar
HUF-Hungarian Forint
ISK-Iceland Krona
INR-Indian Rupee
IDR-Indonesian Rupiah
IQD-Iraqi Dinar
ILS-Israeli Shekel
JMD-Jamaican Dollar
JPY-Japanese Yen
JOD-Jordanian Dinar
KZT-Kazakhstan Tenge
KES-Kenyan Shilling
KRW-Korean Won
KWD-Kuwaiti Dinar
LAK-Lao Kip
LVL-Latvian Lat
LBP-Lebanese Pound
LSL-Lesotho Loti
LRD-Liberian Dollar
LYD-Libyan Dinar
LTL-Lithuanian Lita
MOP-Macau Pataca
MKD-Macedonian Denar
MGF-Malagasy Franc
MWK-Malawi Kwacha
MYR-Malaysian Ringgit
MVR-Maldives Rufiyaa
MTL-Maltese Lira
MRO-Mauritania Ougulya
MUR-Mauritius Rupee
MXN-Mexican Peso
MDL-Moldovan Leu
MNT-Mongolian Tugrik
MAD-Moroccan Dirham
MZM-Mozambique Metical
MMK-Myanmar Kyat
NAD-Namibian Dollar
NPR-Nepalese Rupee
ANG-Neth Antilles Guilder
NZD-New Zealand Dollar
NIO-Nicaragua Cordoba
NGN-Nigerian Naira
KPW-North Korean Won
NOK-Norwegian Krone
OMR-Omani Rial
XPF-Pacific Franc
PKR-Pakistani Rupee
XPD-Palladium Ounces
PAB-Panama Balboa
PGK-Papua New Guinea Kina
PYG-Paraguayan Guarani
PEN-Peruvian Nuevo Sol
PHP-*

*Philippine Peso
XPT-Platinum Ounces
PLN-Polish Zloty
QAR-Qatar Rial
ROL-Romanian Leu
RUB-Russian Rouble
WST-Samoa Tala
STD-Sao Tome Dobra
SAR-Saudi Arabian Riyal
SCR-Seychelles Rupee
SLL-Sierra Leone Leone
XAG-Silver Ounces
SGD-Singapore Dollar
SKK-Slovak Koruna
SIT-Slovenian Tolar
SBD-Solomon Islands Dollar
SOS-Somali Shilling
ZAR-South African Rand
LKR-Sri Lanka Rupee
SHP-St Helena Pound
SDD-Sudanese Dinar
SRG-Surinam Guilder
SZL-Swaziland Lilageni
SEK-Swedish Krona
TRY-Turkey Lira
CHF-Swiss Franc
SYP-Syrian Pound
TWD-Taiwan Dollar
TZS-Tanzanian Shilling
THB-Thai Baht
TOP-Tonga Pa'anga
TTD-Trinidad&Tobago Dollar
TND-Tunisian Dinar
TRL-Turkish Lira
USD-U.S. Dollar
AED-UAE Dirham
UGX-Ugandan Shilling
UAH-Ukraine Hryvnia
UYU-Uruguayan New Peso
VUV-Vanuatu Vatu
VEB-Venezuelan Bolivar
VND-Vietnam Dong
YER-Yemen Riyal
YUM-Yugoslav Dinar
ZMK-Zambian Kwacha
ZWD-Zimbabwe Dollar </blockquote></wsdl:documentation><wsdl:input message="tns:ConversionRateSoapIn"/><wsdl:output message="tns:ConversionRateSoapOut"/></wsdl:operation></wsdl:portType><wsdl:portType name="CurrencyConvertorHttpGet"><wsdl:operation name="ConversionRate"><wsdl:documentation>
Get conversion rate from one currency to another currency
 <u>Differenct currency Code and Names around the world</u> <blockquote> <font face='Verdana' size='1'>AFA-Afghanistan Afghani
ALL-Albanian Lek
DZD-Algerian Dinar
ARS-Argentine Peso
AWG-Aruba Florin
AUD-Australian Dollar
BSD-Bahamian Dollar
BHD-Bahraini Dinar
BDT-Bangladesh Taka
BBD-Barbados Dollar
BZD-Belize Dollar
BMD-Bermuda Dollar
BTN-Bhutan Ngultrum
BOB-Bolivian Boliviano
BWP-Botswana Pula
BRL-Brazilian Real
GBP-British Pound
BND-Brunei Dollar
BIF-Burundi Franc
XOF-CFA Franc (BCEAO)
XAF-CFA Franc (BEAC)
KHR-Cambodia Riel
CAD-Canadian Dollar
CVE-Cape Verde Escudo
KYD-Cayman Islands Dollar
CLP-Chilean Peso
CNY-Chinese Yuan
COP-Colombian Peso
KMF-Comoros Franc
CRC-Costa Rica Colon
HRK-Croatian Kuna
CUP-Cuban Peso
CYP-Cyprus Pound
CZK-Czech Koruna
DKK-Danish Krone
DJF-Dijibouti Franc
DOP-Dominican Peso
XCD-East Caribbean Dollar
EGP-Egyptian Pound
SVC-El Salvador Colon
EEK-Estonian Kroon
ETB-Ethiopian Birr
EUR-Euro
FKP-Falkland Islands Pound
GMD-Gambian Dalasi
GHC-Ghanian Cedi
GIP-Gibraltar Pound
XAU-Gold Ounces
GTQ-Guatemala Quetzal
GNF-Guinea Franc
GYD-Guyana*

*Dollar
HTG-Haiti Gourde
HNL-Honduras Lempira
HKD-Hong Kong Dollar
HUF-Hungarian Forint
ISK-Iceland Krona
INR-Indian Rupee
IDR-Indonesian Rupiah
IQD-Iraqi Dinar
ILS-Israeli Shekel
JMD-Jamaican Dollar
JPY-Japanese Yen
JOD-Jordanian Dinar
KZT-Kazakhstan Tenge
KES-Kenyan Shilling
KRW-Korean Won
KWD-Kuwaiti Dinar
LAK-Lao Kip
LVL-Latvian Lat
LBP-Lebanese Pound
LSL-Lesotho Loti
LRD-Liberian Dollar
LYD-Libyan Dinar
LTL-Lithuanian Lita
MOP-Macau Pataca
MKD-Macedonian Denar
MGF-Malagasy Franc
MWK-Malawi Kwacha
MYR-Malaysian Ringgit
MVR-Maldives Rufiyaa
MTL-Maltese Lira
MRO-Mauritania Ougulya
MUR-Mauritius Rupee
MXN-Mexican Peso
MDL-Moldovan Leu
MNT-Mongolian Tugrik
MAD-Moroccan Dirham
MZM-Mozambique Metical
MMK-Myanmar Kyat
NAD-Namibian Dollar
NPR-Nepalese Rupee
ANG-Neth Antilles Guilder
NZD-New Zealand Dollar
NIO-Nicaragua Cordoba
NGN-Nigerian Naira
KPW-North Korean Won
NOK-Norwegian Krone
OMR-Omani Rial
XPF-Pacific Franc
PKR-Pakistani Rupee
XPD-Palladium Ounces
PAB-Panama Balboa
PGK-Papua New Guinea Kina
PYG-Paraguayan Guarani
PEN-Peruvian Nuevo Sol
PHP-Philippine Peso
XPT-Platinum Ounces
PLN-Polish Zloty
QAR-Qatar Rial
ROL-Romanian Leu
RUB-Russian Rouble
WST-Samoa Tala
STD-Sao Tome Dobra
SAR-Saudi Arabian Riyal
SCR-Seychelles Rupee
SLL-Sierra Leone Leone
XAG-Silver Ounces
SGD-Singapore Dollar
SKK-Slovak Koruna
SIT-Slovenian Tolar
SBD-Solomon Islands Dollar
SOS-Somali Shilling
ZAR-South African Rand
LKR-Sri Lanka Rupee
SHP-St Helena Pound
SDD-Sudanese Dinar
SRG-Surinam Guilder
SZL-Swaziland Lilageni
SEK-Swedish Krona
TRY-Turkey Lira
CHF-Swiss Franc
SYP-Syrian Pound
TWD-Taiwan Dollar
TZS-Tanzanian Shilling
THB-Thai Baht
TOP-Tonga Pa'anga
TTD-Trinidad&Tobago Dollar
TND-Tunisian Dinar
TRL-Turkish Lira
USD-U.S. Dollar
AED-UAE Dirham
UGX-Ugandan Shilling
UAH-Ukraine Hryvnia
UYU-Uruguayan New Peso
VUV-Vanuatu Vatu
VEB-Venezuelan Bolivar
VND-Vietnam Dong
YER-Yemen Riyal
YUM-Yugoslav Dinar
ZMK-Zambian Kwacha
ZWD-Zimbabwe Dollar </blockquote></wsdl:documentation><wsdl:input message="tns:ConversionRateHttpGetIn"/><wsdl:output message="tns:ConversionRateHttpGetOut"/></wsdl:operation></wsdl:portType><wsdl:portType name="CurrencyConvertorHttpPost"><wsdl:operation name="ConversionRate"><wsdl:documentation>
Get conversion rate from one currency to another currency
 <u>Differenct currency Code and Names around the*

*world</u> <blockquote> <font face='Verdana' size='1'>AFA-Afghanistan Afghani
ALL-Albanian Lek
DZD-Algerian Dinar
ARS-Argentine Peso
AWG-Aruba Florin
AUD-Australian Dollar
BSD-Bahamian Dollar
BHD-Bahraini Dinar
BDT-Bangladesh Taka
BBD-Barbados Dollar
BZD-Belize Dollar
BMD-Bermuda Dollar
BTN-Bhutan Ngultrum
BOB-Bolivian Boliviano
BWP-Botswana Pula
BRL-Brazilian Real
GBP-British Pound
BND-Brunei Dollar
BIF-Burundi Franc
XOF-CFA Franc (BCEAO)
XAF-CFA Franc (BEAC)
KHR-Cambodia Riel
CAD-Canadian Dollar
CVE-Cape Verde Escudo
KYD-Cayman Islands Dollar
CLP-Chilean Peso
CNY-Chinese Yuan
COP-Colombian Peso
KMF-Comoros Franc
CRC-Costa Rica Colon
HRK-Croatian Kuna
CUP-Cuban Peso
CYP-Cyprus Pound
CZK-Czech Koruna
DKK-Danish Krone
DJF-Dijibouti Franc
DOP-Dominican Peso
XCD-East Caribbean Dollar
EGP-Egyptian Pound
SVC-El Salvador Colon
EEK-Estonian Kroon
ETB-Ethiopian Birr
EUR-Euro
FKP-Falkland Islands Pound
GMD-Gambian Dalasi
GHC-Ghanian Cedi
GIP-Gibraltar Pound
XAU-Gold Ounces
GTQ-Guatemala Quetzal
GNF-Guinea Franc
GYD-Guyana Dollar
HTG-Haiti Gourde
HNL-Honduras Lempira
HKD-Hong Kong Dollar
HUF-Hungarian Forint
ISK-Iceland Krona
INR-Indian Rupee
IDR-Indonesian Rupiah
IQD-Iraqi Dinar
ILS-Israeli Shekel
JMD-Jamaican Dollar
JPY-Japanese Yen
JOD-Jordanian Dinar
KZT-Kazakhstan Tenge
KES-Kenyan Shilling
KRW-Korean Won
KWD-Kuwaiti Dinar
LAK-Lao Kip
LVL-Latvian Lat
LBP-Lebanese Pound
LSL-Lesotho Loti
LRD-Liberian Dollar
LYD-Libyan Dinar
LTL-Lithuanian Lita
MOP-Macau Pataca
MKD-Macedonian Denar
MGF-Malagasy Franc
MWK-Malawi Kwacha
MYR-Malaysian Ringgit
MVR-Maldives Rufiyaa
MTL-Maltese Lira
MRO-Mauritania Ougulya
MUR-Mauritius Rupee
MXN-Mexican Peso
MDL-Moldovan Leu
MNT-Mongolian Tugrik
MAD-Moroccan Dirham
MZM-Mozambique Metical
MMK-Myanmar Kyat
NAD-Namibian Dollar
NPR-Nepalese Rupee
ANG-Neth Antilles Guilder
NZD-New Zealand Dollar
NIO-Nicaragua Cordoba
NGN-Nigerian Naira
KPW-North Korean Won
NOK-Norwegian Krone
OMR-Omani Rial
XPF-Pacific Franc
PKR-Pakistani Rupee
XPD-Palladium Ounces
PAB-Panama Balboa
PGK-Papua New Guinea Kina
PYG-Paraguayan Guarani
PEN-Peruvian Nuevo Sol
PHP-Philippine Peso
XPT-Platinum Ounces
PLN-Polish Zloty
QAR-Qatar Rial
ROL-Romanian Leu
RUB-Russian Rouble
WST-Samoa Tala
STD-Sao Tome Dobra
SAR-Saudi Arabian Riyal
SCR-Seychelles Rupee
SLL-Sierra Leone Leone
XAG-Silver Ounces
SGD-Singapore Dollar
SKK-Slovak Koruna
SIT-Slovenian Tolar
SBD-Solomon Islands Dollar
SOS-Somali*

*Shilling
ZAR-South African Rand
LKR-Sri Lanka Rupee
SHP-St Helena Pound
SDD-Sudanese Dinar
SRG-Surinam Guilder
SZL-Swaziland Lilageni
SEK-Swedish Krona
TRY-Turkey Lira
CHF-Swiss Franc
SYP-Syrian Pound
TWD-Taiwan Dollar
TZS-Tanzanian Shilling
THB-Thai Baht
TOP-Tonga Pa'anga
TTD-Trinidad&Tobago Dollar
TND-Tunisian Dinar
TRL-Turkish Lira
USD-U.S. Dollar
AED-UAE Dirham
UGX-Ugandan Shilling
UAH-Ukraine Hryvnia
UYU-Uruguayan New Peso
VUV-Vanuatu Vatu
VEB-Venezuelan Bolivar
VND-Vietnam Dong
YER-Yemen Riyal
YUM-Yugoslav Dinar
ZMK-Zambian Kwacha
ZWD-Zimbabwe Dollar</p></blockquote></wsdl:documentation><wsdl:input message="tns:ConversionRateHttpPostIn"/><wsdl:output message="tns:ConversionRateHttpPostOut"/></wsdl:operation></wsdl:portType><wsdl:binding name="CurrencyConvertorSoap" type="tns:CurrencyConvertorSoap"><soap:binding transport="http://schemas.xmlsoap.org/soap/http"/><wsdl:operation name="ConversionRate"><soap:operation soapAction="http://www.webserviceX.NET/ConversionRate" style="document"/><wsdl:input><soap:body use="literal"/></wsdl:input><wsdl:output><soap:body use="literal"/></wsdl:output></wsdl:operation></wsdl:binding><wsdl:binding name="CurrencyConvertorSoap12" type="tns:CurrencyConvertorSoap"><soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/><wsdl:operation name="ConversionRate"><soap12:operation soapAction="http://www.webserviceX.NET/ConversionRate" style="document"/><wsdl:input><soap12:body use="literal"/></wsdl:input><wsdl:output><soap12:body use="literal"/></wsdl:output></wsdl:operation></wsdl:binding><wsdl:binding name="CurrencyConvertorHttpGet" type="tns:CurrencyConvertorHttpGet"><http:binding verb="GET"/><wsdl:operation name="ConversionRate"><http:operation location="/ConversionRate"/><wsdl:input><http:urlEncoded/></wsdl:input><wsdl:output><mime:mimeXml part="Body"/></wsdl:output></wsdl:operation></wsdl:binding><wsdl:binding name="CurrencyConvertorHttpPost" type="tns:CurrencyConvertorHttpPost"><http:binding verb="POST"/><wsdl:operation name="ConversionRate"><http:operation location="/ConversionRate"/><wsdl:input><mime:content type="application/x-www-form-urlencoded"/></wsdl:input><wsdl:output><mime:mimeXml part="Body"/></wsdl:output></wsdl:operation></wsdl:binding><wsdl:service name="CurrencyConvertor"><wsdl:port name="CurrencyConvertorSoap" binding="tns:CurrencyConvertorSoap"><soap:address*

location="http://www.webservicex.net/CurrencyConvertor.asmx"/></wsdl:port><wsdl:port name="CurrencyConvertorSoap12" binding="tns:CurrencyConvertorSoap12"><soap12:address location="http://www.webservicex.net/CurrencyConvertor.asmx"/></wsdl:port><wsdl:port name="CurrencyConvertorHttpGet" binding="tns:CurrencyConvertorHttpGet"><http:address location="http://www.webservicex.net/CurrencyConvertor.asmx"/></wsdl:port><wsdl:port name="CurrencyConvertorHttpPost" binding="tns:CurrencyConvertorHttpPost"><http:address location="http://www.webservicex.net/CurrencyConvertor.asmx"/></wsdl:port></wsdl:service></wsdl:definitions>

- WSDL code of *CurrencyService*:

<wsdl:definitions name="CurrencyService" targetNamespace="http://www.restfulwebservices.net/ServiceContracts/2008/01"><wsdl:types><xsd:schema targetNamespace="http://www.restfulwebservices.net/ServiceContracts/2008/01/Imports"><xsd:import schemaLocation="http://www.restfulwebservices.net/wcf/CurrencyService.svc?xsd=xsd0" namespace="http://www.restfulwebservices.net/ServiceContracts/2008/01"/><xsd:import schemaLocation="http://www.restfulwebservices.net/wcf/CurrencyService.svc?xsd=xsd3" namespace="http://GOTLServices.FaultContracts/2008/01"/><xsd:import schemaLocation="http://www.restfulwebservices.net/wcf/CurrencyService.svc?xsd=xsd1" namespace="http://schemas.microsoft.com/2003/10/Serialization/"/><xsd:import schemaLocation="http://www.restfulwebservices.net/wcf/CurrencyService.svc?xsd=xsd2" namespace="http://www.restfulwebservices.net/DataContracts/2008/01"/></xsd:schema></wsdl:types><wsdl:message name="ICurrencyService_GetConversionRate_InputMessage"><wsdl:part name="parameters" element="tns:GetConversionRate"/></wsdl:message><wsdl:message name="ICurrencyService_GetConversionRate_OutputMessage"><wsdl:part name="parameters" element="tns:GetConversionRateResponse"/></wsdl:message><wsdl:message name="ICurrencyService_GetConversionRate_DefaultFaultContractFault_FaultMessage"><wsdl:part name="detail" element="q1:DefaultFaultContract"/></wsdl:message><wsdl:portType name="ICurrencyService"><wsdl:operation name="GetConversionRate"><wsdl:input wsaw:Action="GetConversionRate" message="tns:ICurrencyService_GetConversionRate_InputMessage"/><wsdl:output wsaw:Action="http://www.restfulwebservices.net/ServiceContracts/2008/01/ICurrencyService

/GetConversionRateResponse"
message="tns:ICurrencyService_GetConversionRate_OutputMessage"/><wsdl:fault
wsaw:Action="http://www.restfulwebservices.net/ServiceContracts/2008/01/ICurrencyService
/GetConversionRateDefaultFaultContractFault" name="DefaultFaultContractFault"
message="tns:ICurrencyService_GetConversionRate_DefaultFaultContractFault_FaultMess
age"/></wsdl:operation></wsdl:portType><wsdl:binding
name="BasicHttpBinding_ICurrencyService" type="tns:ICurrencyService"><soap:binding
transport="http://schemas.xmlsoap.org/soap/http"/><wsdl:operation
name="GetConversionRate"><soap:operation soapAction="GetConversionRate"
style="document"/><wsdl:input><soap:body
use="literal"/></wsdl:input><wsdl:output><soap:body
use="literal"/></wsdl:output><wsdl:fault name="DefaultFaultContractFault"><soap:fault
name="DefaultFaultContractFault"
use="literal"/></wsdl:fault></wsdl:operation></wsdl:binding><wsdl:service
name="CurrencyService"><wsdl:port name="BasicHttpBinding_ICurrencyService"
binding="tns:BasicHttpBinding_ICurrencyService"><soap:address
location="http://www.restfulwebservices.net/wcf/CurrencyService.svc"/></wsdl:port></wsdl
:service></wsdl:definitions>

- List of registered services (*ListOfApps.jsp*):

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@page import="javax.xml.namespace.QName"%>
<%@page import="org.apache.axis2.AxisFault"%>
<%@page import="org.apache.axis2.addressing.EndpointReference"%>
<%@page import="org.apache.axis2.addressing.EndpointReference"%>
<%@page
import="org.apache.axis2.client.Options,org.apache.axis2.transport.http.AxisAdminServlet"
%>
<%@page import="org.apache.axis2.client.Options"%>
<%@page import="org.apache.axis2.rpc.client.RPCServiceClient"%>
 <%@page import="java.util.*,com.DiseaseQsStub,com.DiseaseQsStub.*"%>
<%@page import="org.apache.axis2.engine.AxisConfiguration"%>
<%@ page import="org.apache.axis2.Constants,
            org.apache.axis2.description.AxisService,
            java.util.Collection,
            java.util.HashMap,
            java.util.Iterator"%>
<html>
<head>
```

```
<title>Integrated Interface - List Of Applications</title>
</head>
<body><center>
<table>
<%
if((request.getParameter("name")!=null)){
        session.setAttribute("name",request.getParameter("name"));
}
if((session.getAttribute("newuser")!=null)){
        session.setAttribute("name",session.getAttribute("newuser"));
}
if((session.getAttribute("newuser")!=null) || (session.getAttribute("name")!=null) ){ %>
<%
if(((session.getAttribute("name")!=null)  &&  (session.getAttribute("newuser")==null))  ||
(application.getAttribute("name")!=null)){%>
<tr><td width="900"><jsp:include page="Top.jsp"/></td></tr>
<tr><td  align="right">Welcome  <b><font  color="blue"
size="4"><%=session.getAttribute("name").toString()%>!!</font></b></td></tr>
<tr><td><jsp:include page="ListAvailableApps.jsp"/></td></tr>
<tr height="20"><td height="100"></td></tr>
<tr><td><jsp:include page="SOA.jsp"/></td></tr>
<%}%>
<%if(session.getAttribute("newuser")!=null){%>
<tr><td width="900"><jsp:include page="Top.jsp"/></td></tr>
<tr><td align="right">Registration Success
Welcome  <b><font  color="blue"
size="4"><%=session.getAttribute("newuser").toString()%>!!</font></b></td></tr>
<tr><td><jsp:include page="ListAvailableApps.jsp"/></td></tr>
<tr height="20"><td height="100"></td></tr>
<tr><td><jsp:include page="SOA.jsp"/></td></tr>
<%}%>
<%
}else{
RequestDispatcher rd=request.getRequestDispatcher("Login.jsp");
rd.forward(request,response);
}
        %>
 <tr><td width="900"><jsp:include page="Bottom.jsp"/></td></tr>
</table></center>
```

```
</body>
</html>
```

- List of available services (*ListAvailableApps.jsp*):

```
<htmL>
<head>
<script type="text/javascript">
</script>
<title></title>
</head>
<body>
<div align="left">
<font size="4" color="green">Click to view Available Applications List</font>
<table>
<!--provide server's IP address here -->
<%if((session.getAttribute("newuser")!=null) || (session.getAttribute("name")!=null) ){
if(session.getAttribute("name")!=null){
        //System.out.println(session.getAttribute("name").toString()+"Here");
if(session.getAttribute("name").toString().equals("admin")){ %>
<%//System.out.println("true");%>
<tr><td><a href="http://164.0.4.27:8080/IIS/services/listServices?un=admin"><font
size="4" color="blue">List Of Available Applications</font> </a></td></tr>
<%}else{%>
<tr><td><a href="http://164.0.4.27:8080/IIC/rdCtoS.jsp"><font size="4"
color="blue">List Of Available Applications</font> </a></td></tr>
<%}
}else if(session.getAttribute("newuser")!=null){ %>
<tr><td><a href="http://164.0.4.27:8080/IIC/rdCtoS.jsp"><font size="4"
color="blue">List Of Available Applications</font></a></td></tr>
<%} }%>
</table>
</div>
</body>
</htmL>
```

- Services search (*SOASearch.jsp & SearchAppsType.jsp*):

  SOASearch.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body><center>
<jsp:include page="Top.jsp" />
<form action="SearchAppsType.jsp">
<table>
<tr><td height="30"></td></tr>
<tr><td><label><b><font size="3" color="green">Search By
KeyWord</font></b></label></td></tr>
<tr><td><input type="text" size="70" name="searchbox"/></td></tr>
<tr><td><input type="submit" name="submit" value="Search"/></td></tr>
</table>
</form>
<table>
<tr><td height="30"></td></tr>
<tr><td><a href="http://164.0.4.27:8080/IIC/ListOfApps.jsp">Home</a></td></tr>
</table>
<jsp:include page="Bottom.jsp"/>
</center>
</body>
</html>
```

SearchAppsType.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
   <%@page import="java.sql.*,java.util.*" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Search Application By Type</title>
</head>
<body><center>
```

```
<jsp:include page="Top.jsp"></jsp:include>
<%
String searchText=request.getParameter("searchbox").toString();
%>
<table>
<tr><td height="40"></td></tr>
<tr><td><font color="blue" size="5"> Results from search text :
<%=searchText%></font></td></tr>
</table>
<%
Connection con = null;
boolean flag=false;
 try {
    Class.forName("com.mysql.jdbc.Driver");
    con = DriverManager.getConnection("jdbc:mysql:///iidb","root", "macb96");
                String appname=null;
    Statement stmt=con.createStatement();
                String query="select appname from knowledge_base k, application_details
a where k.appname = a.Application_Name and a.Application_Service_Status = 'Active' and
a.Application_Type = 'V' and (k.keyword like '%"+searchText.toLowerCase()+"%' or
k.description like '%"+searchText.toLowerCase()+"%');";
                 ResultSet rst=stmt.executeQuery(query);
                while(rst.next()){
                 appname=rst.getString(1);
                // System.out.println(appname);
                flag = true;
                %>
                <table>
                <tr><td height="30"></td></tr>
                <%if(appname.equals("DiseaseQs")){ %>
                <tr><td><a                href="http://164.0.4.27:8080/IIC/noc.jsp"><font
color="green" size="5"><%=appname%></font></a></td></tr>
                <%}else{
                        %>
                        <tr><td><a        href="http://164.0.4.27:8080/IIC/sf.jsp"><font
color="green" size="5"><%=appname%></font></a></td></tr>
                <%} %>
                </table>
                <%
```

```
                    }
                    if(!flag){%>
                    <table>
                    <tr><td height="30"></td></tr>
                    <tr><td><font size="4" color="green">Your search</font>-<font size="5" color="blue"><%=searchText%></font>-<font size="4" color="green">did not match any applications. </font></td></tr>
                    </table>
          <%}
 }catch(SQLException e){
           e.printStackTrace();
 }
 %>
 <table>
 <tr><td height="30"></td></tr>
 <tr><td><a href="http://164.0.4.27:8080/IIC/ListOfApps.jsp">Home</a></td></tr>
 </table>
 <jsp:include page="Bottom.jsp"></jsp:include>
 </center>
 </body>
 </html>
```

- Example of web service interfaces, i.e. SunFlow image rendering (*sf.jsp & sfAction.jsp*) and scene file (*cornell_box_jensen.sc*), and survey visualizer (*userInput.jsp & chartShow.jsp*):

  sf.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
    <%@page import="java.sql.*,java.util.Date,java.text.*,com.*" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
 <head><title>Image Renderer</title></head>
<script type="text/javascript">
function checkFileType(){
          var filename=document.uplform.upl.value;
          //alert(filename);
          if(filename.endsWith("sc")){
                    return true;
```

```
        }else
        {
                alert("File extension should be .sc\nOnly Scene files to be Uploaded");
                document.uplform.upl.value=" ";
                document.uplform.upl.focus();
                return false;
        }
}
</script>
 <%
   Connection con = null;
   boolean flag=false;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            con = DriverManager.getConnection("jdbc:mysql:///iidb",
              "root", "macb96");
            Statement stmt=con.createStatement();
                        String s_Name="ImageRenderer";
             String query="select Serviced_Count from service_history where
Service_Name='"+s_Name+"'";
                                ResultSet rst=stmt.executeQuery(query);
                                int Serviced_Count_i=0;
                                if(rst.next()){
                                        Serviced_Count_i
=rst.getInt("Serviced_Count");
                                        Serviced_Count_i=Serviced_Count_i+1;
                                }
                                query="update service_history set
Serviced_Count='"+Serviced_Count_i+"' where Service_Name='"+s_Name+"'";
                                stmt.executeUpdate(query);
                                query="update knowledge_base set
hits='"+Serviced_Count_i+"' where appname='"+s_Name+"'";
                                stmt.executeUpdate(query);
         }catch(SQLException e){
                 e.printStackTrace();
        }
   %>
    <body> <center>
 <jsp:include page="Top.jsp"></jsp:include>
```

```
 <%
  if( request.getAttribute("sfonly")!=null){ %>
        <table><tr><td height="30"></td></tr>
        <tr><td><font color="green" size="4"> File extension should be .sc and Only Scene files to be Uploaded</font></td></tr></table>
 <% } %>
 <form action="sfAction.jsp" method="post" enctype="multipart/form-data" onsubmit="return checkFileType()" name="uplform" >
 <table>
<tr><td>Select a file to be rendered</td><td>:</td><td colspan="2"><input type="file" name="upl"></td></tr>
<tr><td></td><td></td><td align="right"><input type="submit" value="submit" name="imagesubmit" ></td><td align="left"><input type=reset></td></tr>
</table>
 </form>
 <table><tr><td><a href="http://164.0.4.27:8080/IIC/ListOfApps.jsp">Home</a></td></tr></table>
 <jsp:include page="Bottom.jsp"></jsp:include>
 </center>
 </body>
 </html>
```

sfAction.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%@ page import="java.util.List" %>
  <%@ page import="java.util.Iterator" %>
  <%@ page import="java.io.File" %>
  <%@ page import="org.apache.commons.fileupload.servlet.ServletFileUpload"%>
  <%@ page import="org.apache.commons.fileupload.disk.DiskFileItemFactory"%>
  <%@ page import="org.apache.commons.fileupload.*"%>
  <%@page import="com.SunflowGUI"%>
<%@page import="com.ImagePanel"%>
<%@page import="com.ImageRendererStub"%>
<%@page import="com.ImageRendererStub.*"%>
<html>
<body>
```

```
<center>
<table>
  <%
 boolean isMultipart = ServletFileUpload.isMultipartContent(request);
 if (!isMultipart) {
 } else {
            FileItemFactory factory = new DiskFileItemFactory();
            ServletFileUpload upload = new ServletFileUpload(factory);
            List items = null;
            try {
                        items = upload.parseRequest(request);
            } catch (FileUploadException e) {
                        e.printStackTrace();
            }
            Iterator itr = items.iterator();
            while (itr.hasNext()) {
            FileItem item = (FileItem) itr.next();
            if (item.isFormField()) {
            } else {
                        try {
                                    String itemName = item.getName();
                                    File f=new File(itemName);
                                    String fname=f.getName();
                                    //System.out.println(fname);
                                    session.setAttribute("fname",fname);
                                    if(fname.endsWith(".sc")){
                                     File file=new File("C:/Program Files/Apache Software
Foundation/Tomcat 6.0/webapps/IIC/image/",fname);
                                    item.write(file);
                                    try {
                                   Thread.sleep(10000);
                        } catch (InterruptedException e) {
                                   e.printStackTrace();
                        }
                                    SunflowGUI sfgui = null;
                                                try {
                                                sfgui = new SunflowGUI();
                                                sfgui.setVisible(false);
```

```
                                        ImagePanel imp =new
ImagePanel(fname,request,response);
                                        if(sfgui.fileInput("C:/Program Files/Apache Software
Foundation/Tomcat 6.0/webapps/IIC/image/"+fname)){
                                                imp.display();
                                        }
                                        } catch (Exception e) {
                                        if (sfgui != null)
                                                sfgui = null;
                                        e.printStackTrace();
                        }
                }else{
                        RequestDispatcher rd=request.getRequestDispatcher("sf.jsp");
                        request.setAttribute("sfonly","true");
                        rd.forward(request,response);
                }
                                }catch (Exception e) {
                        e.printStackTrace();
                }
                }
  }
}
  %>
   </table>
  </center>
  </body>
```

cornell_box_jensen.sc:

```
image {
  resolution 800 600
  aa 0 2
  filter gaussian
}

trace-depths {
  diff 4
  refl 3
  refr 2
}
```

```
photons {
  caustics 1000000 kd 100 0.5
}

% uncomment this block and comment the following GI block to switch gi engines
/*
gi {
  type irr-cache
  samples 512
  tolerance 0.01
  spacing 0.05 5.0
% comment the following line to use path tracing for secondary bounces
  global   1000000 grid 100 0.75
}
*/

gi {
  type igi
  samples 64        % number of virtual photons per set
  sets 1            % number of sets (increase this to translate shadow boundaries into noise)
  b 0.00003         % bias - decrease this values until bright spots dissapear
  bias-samples 0    % set this >0 to make the algorithm unbiased
}

shader {
  name debug_caustics
  type view-caustics
}

shader {
  name debug_globals
  type view-global
}

shader {
  name debug_gi
  type view-irradiance
}
```

```
%% use these to view the effect of the individual gi components
% override debug_caustics false
% override debug_globals false
% override debug_gi false

camera {
  type pinhole
  eye    0 -205 50
  target 0 0 50
  up     0 0 1
  fov    45
  aspect 1.333333
}

shader {
  name Grey
  type diffuse
  diff 0.7 0.7 0.7
}

shader {
  name Blue
  type diffuse
  diff 0.25 0.25 0.8
}

shader {
  name Red
  type diffuse
  diff 0.8 0.25 0.25
}

shader {
  name Mirror
  type mirror
  refl 0.7 0.7 0.7
}
```

```
shader {
  name Glass
  type glass
  eta 1.6
  color 1 1 1
}

object {
  shader none
  type cornellbox
  corner0 -60 -60 0
  corner1  60  60 100
  left    0.80 0.25 0.25
  right   0.25 0.25 0.80
  top     0.70 0.70 0.70
  bottom  0.70 0.70 0.70
  back    0.70 0.70 0.70
  emit    15 15 15
  samples 32
}

object {
  shader Mirror
  type sphere
  c -30 30 20
  r 20
}

object {
  shader Glass
  type sphere
  c 28 2 20
  r 20
}
```

userInput.jsp:

```
<html>
<head>
<script type="text/javascript">
```

```
function checkform ( form )
{
        <%for(int i=1;i<=Integer.valueOf(session.getAttribute("noc").toString());i++){
        %>
        if (form.c<%=i%>.value == " ") {
                alert( "Please enter a Disease name." );
                form.c<%=i%>.focus();
                return false ;
               }
        if(!(isNaN(form.c<%=i%>.value))){
                alert( "Disease name should be alphanumeric or characters." );
                form.c<%=i%>.value = "";
                form.c<%=i%>.focus();
                return false ;
         }
        <%}%>
        var dnames=new Array();
        var cno=0,di=0,di1=0;
        <%for(int i=1;i<=Integer.valueOf(session.getAttribute("noc").toString());i++){%>
        dnames[di]=form.c<%=i%>.value;
        di++;
        <%} %>di=0;
        <%for(int i=1;i<=Integer.valueOf(session.getAttribute("noc").toString());i++){%>
                var name=dnames[di];
                cno = 0;
                for(di1=0;di1<dnames.length;di1++){
                if(name===dnames[di1]){
                        cno=cno+1;
                        if(cno>1){
                                alert( "Diseases name should not be Equal." );
                           form.c<%=i%>.focus();
                           return false ;
                        }
                }
        }di=di+1;
        <%}%>
        <%for(int p=1;p<=Integer.valueOf(session.getAttribute("noc").toString());p++){
                for(int j=1;j<=4;j++){%>
                if((form.c<%=p%><%=j%>.value == "")){
```

```
                alert( "Please enter Quater values" );
                form.c<%=p%><%=j%>.value = "";
                form.c<%=p%><%=j%>.focus();
                return false;
        }
        if(isNaN(form.c<%=p%><%=j%>.value)){
                alert( "Please enter numbers only" );
                form.c<%=p%><%=j%>.value = "";
                form.c<%=p%><%=j%>.focus();
                return false;
        }
        if(form.c<%=p%><%=j%>.value==0){
                alert( "Quater values should not be zero(0)" );
                form.c<%=p%><%=j%>.value = "";
                form.c<%=p%><%=j%>.focus();
                return false;
        }
        re = /[A-Z]/;
         if(re.test(form.c<%=p%><%=j%>.value))
         {
        alert("Quater values must NOT contain charaters!");
        form.c<%=p%><%=j%>.value = "";
          form.c<%=p%><%=j%>.focus();
        form.pass1.focus();
        return false;
        }
         re = /[a-z]/;
         if(re.test(form.c<%=p%><%=j%>.value))
         {
        alert("Quater values must NOT contain charaters!");
        form.c<%=p%><%=j%>.value = "";
          form.c<%=p%><%=j%>.focus();
        form.pass1.focus();
        return false;
        }
        <%}%>
        <%for(int l=1;l<=4;l++){%>
        if((form.d<%=p%><%=l%>.value=="")                                ||
((isNaN(form.c<%=p%><%=l%>.value)))){
```

```
                    alert( "Please enter Quater values" );
                    form.d<%=p%><%=l%>.value = "";
                    form.d<%=p%><%=l%>.focus();
                    return false;
                }
                if((form.d<%=p%><%=l%>.value=="")                    ||
((isNaN(form.c<%=p%><%=l%>.value)))){
                    alert( "Quater values should not be zero(0)" );
                    form.d<%=p%><%=l%>.value = "";
                    form.d<%=p%><%=l%>.focus();
                    return false;
                }
                var re = /^\w+$/;
                if(!re.test(form.d<%=p%><%=l%>.value))
                      {
                    alert("Quater values must contain only numbers !");
                      form.d<%=p%><%=l%>.value = "";
                            form.d<%=p%><%=l%>.focus();
                 return false;
                 }
                re = /[A-Z]/;
                 if(re.test(form.d<%=p%><%=l%>.value))
                 {
                alert("Quater values must NOT contain charaters!");
                form.d<%=p%><%=l%>.value = "";
                 form.d<%=p%><%=l%>.focus();
                return false;
                }
                 re = /[a-z]/;
                 if(re.test(form.d<%=p%><%=l%>.value))
                 {
                alert("Quater values must NOT contain charaters!");
                form.d<%=p%><%=l%>.value = "";
                 form.d<%=p%><%=l%>.focus();
                return false;
                }
                <%}%>
            <%}%>
             return true;
```

```
                    }
</script>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Integrated Interface</title>
</head>
<body>
<center>
<jsp:include page="Top.jsp"></jsp:include>
<h3><b><font color="blue">Disease comparison for the year 2008 and 2009</font></b></h3>
<table><tr><td><%if(request.getAttribute("dname")!=null){ %>Diseases names should NOT be same...please try again<%} %></td></tr></table>
<%
int no=0;
if(session.getAttribute("noc")!=null){
        Integer in=Integer.valueOf(session.getAttribute("noc").toString());
        no=in.intValue();
}
        int c=1;%>
<form action="DiseasesShow.jsp" name="cn" method="POST" onsubmit="return checkform(this);">
<table><%for(int k=1;k<=no;k++){
%>
        <tr align = "justify" bgcolor="lightblue">
<td align="right"><b>Disease Name </b></td><td align="left" colspan="2"><input type="text" name="c<%=k%>" id="cname"/></td></tr>
<tr>
<td colspan="2"><table>
<tr><th align="center">Year</th><th>2008</th><th>2009</th></tr>
<tr bgcolor="lightgreen">
<%for(int r=1;r<=4;r++){ %>
<td align="right" width="20" bgcolor="lightgreen"><b>Quarter<%=r %></b></td><td bgcolor="lightgreen">
<input type="text" name="c<%=c%><%=r%>" size="7" id="ncone"></td><td bgcolor="lightgreen"><input type="text" name="d<%=c%><%=r%>" size="7" id="nctwo"></td>
</tr><%} %>
<tr><td> </td></tr>
</table></td></tr>
```

```
<%c++;} %>
<tr><td colspan="4"><input type="submit" name="cn" value="submit"></td></tr>
</table>
<table><tr><td><a href="noc.jsp">Back</a></td><td><a href="ListOfApps.jsp">Home</a></td></tr></table>
</form>
<jsp:include page="Bottom.jsp"></jsp:include>
</center>
</body>
</html>
```

chartShow.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<%@ page import="java.awt.*,java.util.Random"%>
<%@ page import="java.io.*"%>
<%@ page import="org.jfree.chart.*"%>
<%@ page import="org.jfree.chart.axis.*"%>
<%@ page import="org.jfree.chart.entity.*"%>
<%@ page import="org.jfree.chart.labels.*"%>
<%@ page import="org.jfree.chart.plot.*"%>
<%@ page import="org.jfree.chart.renderer.category.*"%>
<%@ page import="org.jfree.chart.urls.*"%>
<%@ page import="org.jfree.data.category.*"%>
<%@ page import="org.jfree.data.general.*"%>
<%@ page import="org.jfree.ui.*"%>
<%@page import="java.util.ArrayList"%>
<%@page import="java.util.*,com.DiseaseQsStub,com.DiseaseQsStub.*"%>
<html>
<head>
<title>Integrated Interface - Disease Death Survey Visualizer</title>
</head>
<%
        HttpSession ses = request.getSession();
        int no = 0;
        Integer in = null;
```

```
if(ses.getAttribute("noc")!=null){
        in = Integer.valueOf((ses.getAttribute("noc")).toString());
        no = in.intValue();
        }
        String[] dnames =new String[no];
        for(int i=0;i<no;i++){
                dnames[i]=request.getParameter("c"+""+(i+1));
        }int cno=0;
        for(int i=0;i<no;i++){
                String name=dnames[i];
                for(i=0;i<no;i++){
                if(name.equals(dnames[i])){
                        cno=cno+1;
                        if(cno>1){
                        request.setAttribute("dname","dname");
                        RequestDispatcher
rd=request.getRequestDispatcher("UserInput.jsp");
                        rd.forward(request,response);
                        }
                }
                }
        }
        Color[] c=new Color[no];
        int[] colors=new int[3];
        int max=255;
        int min=0;
        for(int j=0;j<no;j++){
        for(int i=0;i<3;i++){
                colors[i] = (int) (Math.random() * (max - min + 1) ) + min;
                if(i>1){
                        if(colors[0]==colors[1]){
                                i--;
                                continue;
                        }
                }
                }c[j] = new Color(colors[0],colors[1], colors[2]);
        }
        int noc=no*4;
        int[] Qs2008=new int[noc];
```

```
int[] Qs2009=new int[noc];
int[] Qsper=new int[noc];
int k=0;
for (int i = 1; i <=no; i++) {
                for(int j=1;j<=4;j++){
        Qs2008[k]=Integer.parseInt(request.getParameter("c"+""+(i)+""+(j)));
                        k++;
                }
        }
        k=0;
        for (int i = 1; i <=no; i++) {
                for(int j=1;j<=4;j++){
        Qs2009[k]=Integer.parseInt(request.getParameter("d"+""+(i)+""+(j)));
                        k++;
                }
        }
        String[] quarter = { "First quarter", "Second quarter",
                        "Third quarter", "Fourth quarter" };
        //create the dataset...
        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
                k=0;
                for (int r = 0; r < no; r++) {
                                for (int i = 0; i < quarter.length; i++) {
                                dataset.addValue(Qs2008[k], dnames[r],
                                                quarter[i]);
                                k++;
                }
        }
        final JFreeChart chart = ChartFactory.createBarChart(
                        "Deaths in 2008", // chart title
                        "Quarter", // domain axis label
                        "No. of Deaths", // range axis label
                        dataset, // data
                        PlotOrientation.VERTICAL, // orientation
                        true, // include legend
                        true, // tooltips?
                        false // URLs?
                        );
        // NOW DO SOME OPTIONAL CUSTOMISATION OF THE CHART...
```

```
                // set the background color for the chart...
                chart.setBackgroundPaint(Color.white);
                final CategoryPlot plot = chart.getCategoryPlot();
                plot.setBackgroundPaint(Color.LIGHT_GRAY);
                plot.setDomainGridlinePaint(Color.white);
                plot.setRangeGridlinePaint(Color.blue);
                // get a reference to the plot for further customisation...
                // set the range axis to display integers only...generating numbers in y axis
                final NumberAxis rangeAxis = (NumberAxis) plot.getRangeAxis();
                rangeAxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits());
                // disable bar outlines...
                final BarRenderer renderer = (BarRenderer) plot.getRenderer();
                renderer.setDrawBarOutline(false);
                // set up gradient paints for series...
                for(int i=0;i<no;i++){
        renderer.setSeriesPaint(i, c[i]);
        }
                final CategoryAxis domainAxis = plot.getDomainAxis();
                domainAxis.setCategoryLabelPositions(CategoryLabelPositions
                                .createUpRotationLabelPositions(1));
                try {
                        final ChartRenderingInfo info = new ChartRenderingInfo(
                                        new StandardEntityCollection());
                        final File file1 = new File("C:/Program Files/Apache Software
Foundation/Tomcat 6.0/webapps/IIC/image/3dbarchart.png");
                        ChartUtilities.saveChartAsPNG(file1, chart, 600, 400, info);
                } catch (Exception e) {
                        out.println(e);
                }
                final DefaultCategoryDataset dataset2 = new DefaultCategoryDataset();
                k=0;
                for (int r = 0; r < no; r++) {
                                for (int i = 0; i < quarter.length; i++) {

                                dataset2.addValue(Qs2009[k], dnames[r],
                                                quarter[i]);
                        k++;
                }
        }
```

```
final JFreeChart chart2 = ChartFactory.createBarChart(
                        "Deaths in 2009", // chart title
                        "Quarter", // domain axis label
                        "No. of Deaths", // range axis label
                        dataset2, // data
                        PlotOrientation.VERTICAL, // orientation
                        true, // include legend
                        true, // tooltips?
                        false // URLs?
                        );
// NOW DO SOME OPTIONAL CUSTOMISATION OF THE CHART...
// set the background color for the chart...
chart2.setBackgroundPaint(Color.white);
final CategoryPlot plot2 = chart2.getCategoryPlot();
plot2.setBackgroundPaint(Color.LIGHT_GRAY);
plot2.setDomainGridlinePaint(Color.white);
plot2.setRangeGridlinePaint(Color.blue);
// get a reference to the plot for further customisation...
// set the range axis to display integers only...generating numbers in y axis
final NumberAxis rangeAxis2 = (NumberAxis) plot2.getRangeAxis();
rangeAxis2.setStandardTickUnits(NumberAxis.createIntegerTickUnits());
// disable bar outlines...
final BarRenderer renderer2 = (BarRenderer) plot2.getRenderer();
renderer2.setDrawBarOutline(false);
// set up gradient paints for series...
for(int i=0;i<no;i++){
renderer2.setSeriesPaint(i, c[i]);
}
final CategoryAxis domainAxis2 = plot2.getDomainAxis();
domainAxis2.setCategoryLabelPositions(CategoryLabelPositions
                        .createUpRotationLabelPositions(1));
try {
        final ChartRenderingInfo info2 = new ChartRenderingInfo(
                                new StandardEntityCollection());
        final File file12 = new File("C:/Program Files/Apache Software
Foundation/Tomcat 6.0/webapps/IIC/image/3dbarchart2.png");
        ChartUtilities.saveChartAsPNG(file12, chart2, 600, 400, info2);
} catch (Exception e) {
        out.println(e);
```

```
}
final DefaultCategoryDataset dataset3 = new DefaultCategoryDataset();
                    DiseaseQsStub qsop=new DiseaseQsStub();
                    GetSubs subs=new GetSubs();
                    subs.setQs2008(Qs2008);
                    subs.setQs2009(Qs2009);
                    subs.setN(no);
                     GetSubsResponse subresp=qsop.getSubs(subs);
                     int diff[]=subresp.get_return();
                     //System.out.println(diff.length);
                    k=0;
                    for (int r = 0; r < no; r++) {
                    for (int i = 0; i < quarter.length; i++) {
                              dataset3.addValue(diff[k], dnames[r],
                                                  quarter[i]);
                              k++;
                    }
}
final JFreeChart chart3 = ChartFactory.createBarChart(
                    "Deaths comparison in 2008 and 2009", // chart title
                    "Quarter", // domain axis label
                    "No. of Deaths", // range axis label
                    dataset3, // data
                    PlotOrientation.VERTICAL, // orientation
                    true, // include legend
                    true, // tooltips?
                    false // URLs?
                    );
// NOW DO SOME OPTIONAL CUSTOMISATION OF THE CHART...
// set the background color for the chart...
chart3.setBackgroundPaint(Color.white);
final CategoryPlot plot3 = chart3.getCategoryPlot();
plot3.setBackgroundPaint(Color.LIGHT_GRAY);
plot3.setDomainGridlinePaint(Color.white);
plot3.setRangeGridlinePaint(Color.blue);
// get a reference to the plot for further customisation...
// set the range axis to display integers only...
final NumberAxis rangeAxis3 = (NumberAxis) plot3.getRangeAxis();
rangeAxis3
```

```
                    .setStandardTickUnits(NumberAxis.createIntegerTickUnits());
        // disable bar outlines...
        final BarRenderer renderer3 = (BarRenderer) plot3.getRenderer();
        renderer3.setDrawBarOutline(false);
        // set up gradient paints for series...
        for(int i=0;i<no;i++){
        renderer3.setSeriesPaint(i, c[i]);
        }
        final CategoryAxis domainAxis3 = plot3.getDomainAxis();
        domainAxis3.setCategoryLabelPositions(CategoryLabelPositions
                    .createUpRotationLabelPositions(1));
        try {
                final ChartRenderingInfo info3 = new ChartRenderingInfo(
                            new StandardEntityCollection());
                final File file13 = new File("C:/Program Files/Apache Software
Foundation/Tomcat 6.0/webapps/IIC/image/3dbarchart3.png");
                ChartUtilities.saveChartAsPNG(file13, chart3, 600, 400, info3);
        } catch (Exception e) {
                out.println(e);
        }
        %>
<body><center>
<table>
        <tr>
                <td colspan="3"><IMG SRC="image/3dbarchart.png" WIDTH="600"
                        HEIGHT="400"                                    BORDER="1"
USEMAP="#chart"></td><td></td>
        </tr>
        <tr>
                <td colspan="3"><IMG SRC="image/3dbarchart2.png" WIDTH="600"
                        HEIGHT="400"                                    BORDER="1"
USEMAP="#chart"></td><td></td>
        </tr>
        <tr>
                <td colspan="3"><IMG SRC="image/3dbarchart3.png" WIDTH="600"
                        HEIGHT="400" BORDER="1" USEMAP="#chart"></td>
        </tr>
        <tr>
                <td colspan="3"> </td>
```

```
</tr>
<tr>
        <td colspan="3"> </td>
</tr>
<tr>
        <td colspan="3"></td>
</tr>
<tr>
        <td colspan="3"></td>
</tr>
<tr>
        <td colspan="3"></td>
</tr>
<tr><td>
         <table  bordercolor="yellow">
         <tr><th bgcolor="orange" colspan="5">Disease Death Rate for the year 2009</th></tr>
        <tr        bgcolor="lightgreen"><th>Disease     Name</th><th>Quarter 1</th><th>Quarter 2</th><th>Quarter 3</th><th>Quarter 4</th></tr>
        <%k=0;
        DiseaseQsStub qsper=new DiseaseQsStub();
        GetPers per=new GetPers();
        per.setQs2008(Qs2008);
        per.setQs2009(Qs2009);
        per.setN(no);
        GetPersResponse perresp=qsper.getPers(per);
        float qper[] = perresp.get_return();
        for(int r=0;r<no;r++){
        //for(int col=0;col<no;col++){
        %><tr >
        <td bgcolor="lightgreen" width="100"><%=(dnames[r])%></td>
<%      for(int i=0;i<4;i++){
                %>
                <td bgcolor="lightgreen"  width="120"><%=(qper[k])%>
                <%if(diff[k]==0){%> No                Change<%}else if(diff[k]>0){%>Up<%}else{%>Down<%} %></td>
                <%k++;}%></tr>
<%      //}
        }
```

```
%></table>
        </td></tr>
        <tr align="center"><td> </td></tr>
    </table>
        <table><tr align="center"><td ><a
href="UserInput.jsp">|Back</a></td><td><a href="ListOfApps.jsp">|Go
Home|</a></td></tr>
</table></center>
</body>
</html>
```