A Framework for Secure Management of Web Services (SMaWS) in Enterprise Application Integration

Der Fakultät für Ingenieurwissenschaften der

Universität Duisburg-Essen

zur Erlangung des akademischen Grades eines

Doktor der Ingenieurwissenschaften (Dr. –Ing.)

genehmigte Dissertation von

Robinson Fornge, Fornge, M.Sc.

aus Akum; Kamerun

Referent: Prof. Dr. –Ing. Alex Hunger

Korreferent: Prof. Dr. Rainer Unland

Tag der mündlichen Prüfung: 18.01.2008

TABLE OF CONTENTS

ABSTRACT **V**

TABLE OF FIGURES **VII**

LIST OF TABLES **IX**

ACKNOWLEDGMENTS **X**

LIST OF ABBREVIATIONS **XII**

CHAPTER 1 INTRODUCTION **1**

1.1 MOTIVATION 1

1.2 WHY DO WE NEED TO MONITOR AND MANAGE WEB SERVICES? 3

1.3 WEB SERVICE MONITORING AND MANAGEMENT CHALLENGES 4

1.4 RESEARCH GOALS 6

1.5 DISSERTATION OUTLINE 7

CHAPTER 2 BACKGROUND INFORMATION **9**

2.1 WHAT IS ENTERPRISE APPLICATION INTEGRATION? 9

2.2 WEB SERVICES 12

2.2.1 What is a Web Service? *12*

2.2.2 Service Oriented Architectures and Web Services *13*

2.2.3 Web Service Architecture *14*

2.3 EAI AND WEB SERVICES 18

2.4 THE USE OF WEB SERVICE WITHIN ENTERPRISES 19

2.4.1 Company Overview *19*

2.4.2 An example of Web Service Infrastructure within pharmaceutical company *20*

2.4.3 Challenges encounter with the new integration solution – Web Service *23*

2.5 WEB SERVICE QUALITY OF SERVICE (QOS) 24

2.5.1 QoS Overview *24*

2.5.2 Monitoring the QoS for Web Services *25*

2.5.2.1 Reliability 25

2.5.2.2 Performance 27

2.5.2.3 Availability 28

2.5.2.4 Safety 28

2.5.2.5 Security 29

CHAPTER 3 MONITORING AND MANAGEMENT OF WEB SERVICES: *REQUIREMENT ANALYSIS* **30**

3.1 INTRODUCTION 30
3.1.1 SMaWS Overview *30*
3.1.2 Use Case Overview *32*
3.2 REQUIREMENT ANALYSIS 33
3.2.1 Description *33*
3.2.2 Usage Scenarios *35*
3.2.2.1 Use case - Global View 36
3.2.2.2 Use Case Monitoring of Web Services View 37
3.2.2.3 Use Case – Management of Web Services 40
3.2.2.4 Use Case – System Access 44
3.2.2.5 Use Case – User Management 47
3.3 OTHER FUNCTIONAL GOALS 49

CHAPTER 4 SMAWS INFRASTRUCTURE 50

4.1 SMAWS ARCHITECTURE 50
4.1.1 Web Service management approach *50*
4.1.2 SMaWS Architectural Model *51*
4.1.3 SMaWS Subsystem *53*
4.1.3.1 SMaWS Agent Subsystem 54
4.1.3.2 SMaWS Registry Subsystem 56
4.1.3.3 SMaWS Manager Subsystem 57
4.2 SMAWS SECURITY ARCHITECTURE 60
4.2.1 Design consideration *60*
4.2.2 Security architecture *61*

CHAPTER 5 SMAWS IMPLEMENTATION AND USAGE 64

5.1 JMX OVERVIEW 64
5.2 MONITORING WEB SERVICE PERFORMANCE 65
5.2.1 Web Service QoS Metrics *66*
5.2.2 Performance Graph *67*
5.3 SMAWS MANAGER APPLICATION 69
5.4 MANAGED WEB SERVICES AND HOST SYSTEM 70
5.5 SMAWS AGENT IMPLEMENTATION 72
5.5.1 Agent module *77*
5.5.2 SMaWS Agent Address. *79*
5.6 SMAWS SECURITY IMPLEMENTATION 82
5.7 USAGE EXAMPLE 86
5.7.1 Web Service Monitoring *87*

5.7.2 Web Service Management *91*

CHAPTER 6 EXPERIMENTAL EVALUATION OF SMAWS **98**

6.1 OVERVIEW 98

6.2 TESTBED NETWORK 98

6.3 EXPERIMENT, RESULTS AND DISCUSSION 101

6.3.1 How large are the memory usage of SMaWS framework? *102*

6.3.2 What is the difference in average response time caused by SMaWS framework on monitored Web Services? *102*

6.3.3 Discussion *104*

CHAPTER 7 RELATED WORKS **106**

7.1 WEB SERVICE MANAGEMENT FRAMEWORKS 106

7.1.1 Farrel and Kreger Web Service management Approaches *106*

7.1.2 Web Service Level Agreement (WSLA) Framework *108*

7.1.3 Web Service Offering Language (WSOL) Framework *113*

7.1.4 Web Service Distributed Management (WSDM) *117*

7.1.5 Web Service Management (WSMN) *118*

7.2 WEB SERVICE MANAGEMENT COMMERCIAL PRODUCTS 120

7.2.1 Actional's Web Service Management products *120*

7.2.2 Computer Associates Unicenter ® Web Services Distributed Management *121*

7.2.3 AmberPoint's SOA Management System *122*

7.2.4 HP OpenView Service Oriented Architecture (SOA) Manager *122*

CHAPTER 8 CONCLUSION AND FUTURE WORKS **124**

8.1 DISCUSSION 124

8.2 ADVANTAGE AND DISADVANTAGES OF SMAWS 125

8.2.1 Advantages *125*

8.2.2 Disadvantages *126*

8.3 SUMMARY OF CONTRIBUTION 126

8.4 FUTURE WORKS 127

REFERENCE **128**

ABSTRACT

This dissertation addresses challenges currently faced by enterprises that have embraced the new technology called Web Service in order to reduce the cost of enterprise application integration (EAI) as well as improve operational efficiency of their mission-critical business processes. The nature of Web Service introduces new challenges such as dependency among applications, and a failure in one application can lead to a failure in other dependent applications. Such challenges have led to a growing need for enterprises to confront Web Service monitoring and management issues as a priority.

As a solution, this dissertation proposes a SMaWS (Secure Management of Web Services) infrastructure for secure monitoring and management of Web Services. Its goals are to provide deeper visibility into Web Service runtime activities as compared to currently Web Service management tools; access to information about the Quality of Service (QoS) of these Web Services; and a unified monitoring environment for Web Services deployed across enterprise business units. This enables an earlier detection of poor performance problem in each interdependent Web Service, which would lead to a faster diagnose and fixing of possible performance issue, and thus maximize availability.

This dissertation describes the requirements analysis for monitoring and management of Web Services across an enterprise environment. It describes the architecture and design of the SMaWS infrastructure proposed for secure monitoring and management of Web Service.

The proposed SMaWS framework enables the instrumentation of existing and newly developed Web Service applications, and extracts Web Service performance statistics. It determines Web Service identity, reliability, availability, security, usage, and license used by Web Service consumers to access a given service.

This dissertation describes the SMaWS Repository and Security concepts that are proposed to address the challenges faced by most distributed architectures to enable the client applications determine the location of the server ("bootstrapping problem"), and at the same time ensuring both the integrity and confidentiality of parties involved.

Finally, this dissertation presents a prototype implementation of SMaWS Manager Application and Sample SMaWS Web Service applications. The experimental results obtained, in terms of overhead induced by the SMaWS framework on the monitored Web Service applications, demonstrate the feasibility of the SMaWS infrastructure.

TABLE OF FIGURES

FIGURE 2:1 POINT-TO-POINT ENTERPRISE APPLICATION INTEGRATION. ---------- 10
FIGURE 2:2 ENTERPRISE APPLICATION INTEGRATION SOLUTION ---------- 11
FIGURE 2:3 WEB SERVICE ARCHITECTURE ---------- 16
FIGURE 2:4 WEB SERVICE STACK (INSPIRED BY [10]) ---------- 16
FIGURE 2:5 EXAMPLE OF A WEB SERVICE'S INFRASTRUCTURE WITHIN A PHARMACEUTICAL COMPANY ------ 21
FIGURE 3:1 SMAWS OVERVIEW WITHIN AN ENTERPRISE ENVIRONMENT ---------- 32
FIGURE 3:2 SYSTEM USERS ---------- 35
FIGURE 3:3 USE CASE –GLOBAL VIEW ---------- 36
FIGURE 3:4 USE CASE - MONITORING OF WEB SERVICES ---------- 38
FIGURE 3:5 USER CASE – MANAGEMENT OF WEB SERVICES ---------- 41
FIGURE 3:6 USE CASE – SYSTEM ACCESS ---------- 44
FIGURE 3:7 USE CASE – USER MANAGEMENT ---------- 47
FIGURE 4:1 SMAWS ARCHITECTURAL MODEL ---------- 52
FIGURE 4:2 MAIN SMAWS SUBSYSTEM DEPENDENCE VIEW ---------- 54
FIGURE 4:3 SMAWS AGENT COMPONENT VIEW ---------- 55
FIGURE 4:4 SMAWS MANAGER COMPONENT VIEW ---------- 58
FIGURE 4:5 SMAWS SECURITY ARCHITECTURE ---------- 62
FIGURE 5:1 SERVICE IS AVAILABLE (UP) ---------- 67
FIGURE 5:2 WEB SERVICE IS FAILING ---------- 68
FIGURE 5:3 WEB SERVICE FAILED (DOWN) ---------- 68
FIGURE 5:4 MANAGED WEB SERVICES AND HOST SYSTEM ---------- 71
FIGURE 5:5 OVERVIEW OF SMAWS AGENT CLASS DIAGRAM ---------- 74
FIGURE 5:6 EXAMPLES OF MANAGEABLE OPERATIONS FOR THE MONITORING AND MANAGEMENT OF WEB SERVICES. ---------- 77
FIGURE 5:7 UML SEQUENTIAL DIAGRAM OF THE INITIALIZATION PROCESS OF THE AGENT MODULE --------- 78
FIGURE 5:8 OVERVIEW OF SMAWS SECURITY MODULES ---------- 83
FIGURE 5:9 LOGIN PAGE ---------- 87
FIGURE 5:10 WEB SERVICE MONITORING PAGE ---------- 89
FIGURE 5:11 WEB SERVICE EXTERNAL RESOURCE STATUS ---------- 91
FIGURE 5:12 QUALITY OF SERVICE (QOS) MANAGEMENT VIEW FOR THE CUSTOMER MANAGEMENT WEB SERVICE ---------- 92
FIGURE 5:13 USAGE ANALYSIS VIEW FOR CUSTOMER MANAGEMENT WEB SERVICE ---------- 93

FIGURE 5:14 LICENSE ANALYSIS VIEW FOR A THE CUSTOMER MANAGEMENT WEB SERVICE ---------- 94
FIGURE 5:15 WEB SERVICE CONFIGURATION VIEW FOR THE CUSTOMER MANAGEMENT WEB SERVICE. ----- 95
FIGURE 5:16 WEB SERVICE TESTING VIEW FOR THE LOGBOOK WEB SERVICE ---------- 96
FIGURE 5:17 USER MANAGEMENT VIEW FOR LOGBOOK WEB SERVICE ---------- 97
FIGURE 6:1 TESTBED - NETWORK FOR MEASURING THE OVERHEAD CAUSED BY MONITORING WEB SERVICES BASED ON SMAWS FRAMEWORK ---------- 100
FIGURE 6:2 EXPERIMENTAL RESULTS OF WEB SERVICE AVERAGE RESPONSE TIME ---------- 104
FIGURE 7:1 WSLA FRAMEWORK ---------- 109
FIGURE 7:2 WEB SERVICE MANAGEMENT NETWORK [28] ---------- 119

LIST OF TABLES

TABLE 1 WEB SERVICE QOS METRIC---------- 66
TABLE 2 EXPLANATION OF THE SMAWS WEB AGENT IMPLEMENTATION CLASSES AND INTERFACES -------- 75
TABLE 3 DESCRIPTION OF IMPLEMENTED COMPONENTS IN THE SMAWS SECURITY MODULE---------- 84
TABLE 4, DESCRIPTION OF TESTBED NETWORK INFRASTRUCTURE FOR THE MEASUREMENT OF THE OVERHEAD CAUSED BY THE SMAWS FRAMEWORK. ----------100
TABLE 5 RESULT OF THE EXPERIMENT MEASURING THE ADDITIONAL MEMORY USAGE CAUSED BY SMAWS FRAMEWORK----------102
TABLE 6 RESULT OF THE EXPERIMENT MEASURING THE ADDITIONAL AVERAGE RESPONSE TIME CAUSED BY SMAWS FRAMEWORK----------104

ACKNOWLEDGMENTS

I wish to express sincere appreciation to my four academic advisors: Prof. Dr. Ing. Axel Hunger, Dr. Christian Pütter, Dr. Weider Dietmar and Dr. Stefan Werner for their assistance throughout my thesis. I have had a great fortune to get to know and interact with them.

I would like to thank the entire executive at Bayer Business Service GmbH for offering me the opportunity to do my research with them. This opportunity has been a rewarding experience that has broadened my horizons. I am grateful for the company support—both financial and educational. I also need to thank my co-workers in particular Ronald Brill, Christina Bierwirth, Dr. Holger Schimanski, and Joachim Zobel, all of whom provided assistant to me throughout my work in the company.

I am grateful to Dr. Christian Pütter for his constant support and encouragement. His talent, broad interest and his intellectual curiosity and intensity, have always served as a source of motivation.

I am grateful to Dr. Weider Dietmar for his advice and support during the final year of my research.

I am grateful to Dr. Ing Stefan Werner, for his advice and encouragement throughout the thesis work.

I am also grateful to Prof. Dr. Rainer Unland, for his engagement in my thesis work.

I would like to thank all the staff and colleagues at the Department of Computer Engineering for their support, especially Mrs. Elvira Laufenburg, Dipl.-Ing. Frank Schwarz, Dr.-Ing. Chinh Phan Cong, Dipl.-Math. Kerstin Luck, Dr.-Ing. Stefan Freinatis, Mr. Bernd Holzke, and Dipl.-Ing. Joachim Zumbraegel.

I also need to thank my fellow students: Karen Tamrazyan, Geoffry Erlangga, Sumelong Ivo Ntungwe, Victoria Kamukara, Satria Haikal, Nickson Nwahiri, and Achiri Sama Teyha. Interacting and supervising their thesis had been a rewarding experience.

Finally, I want to express my deepest appreciation for my parents Mr. Clement Fontashi, and Mrs Odilia Shuri, my sister Divinia Singwa, and my fiancée Helen Tanjong for being a constant source of loving and encouragement throughout these years. All but the least, I dedicate my thesis to my late uncle Mr. Peter Fru Sama who was my main sponsor of my Master degree in Germany.

LIST OF ABBREVIATIONS

A2A	Application -2-Application
API	Application Programming Interface
Axis	Apache eXtensible Interaction System
B2B	Business-2-Business
CA	Computer Associate
CIM	Common Information Model
CORBA	Common Object Request Broker
D/COM	Distributed Component Object Model
DMTF	Distributed Management Task Force
EAI	Enterprise Application Integration
FTP	File Transfer Protocol
HP	Hewlett-Packard Company
HTTP	Hypertext Transport Protocol
IBM	International Business Machines Corporation
IT	Information Technology
IETF	Internet Engineering Task Force
JAAS	Java Authenitcation Authorization Service
JCE	Java Cryptography Extension
JMS	Java Message Service
JMX	Java Management Extensions
JVM	Java Virtual Machine
MIB	Management Information Base
MOM	Message Oriented Middleware
OMI	Open Management Interface
OASIS	Organization for the Advancement of Structural Information Standard
QoS	Quality of Service
RPC	Remote Procedure Calls

SLA	Service Level Agreement
SLO	Service Level Objective
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SNMP	Simple Network Management Protocol
UDDI	Universal Description, Discovery, and Integration
URI	Uniform Resource Identifier
UML	Unified Modeling Language
W3C	World Wide Web Consortium
WBEM	Web Based Enterprise Management
WSDM	Web Service Distribution Management (OASIS)
WSLA	Web Service Level Agreement (IBM)
WSMN	Web Services Management Network
WSTK	Web Service Tool Kit (IBM)
WSOI	Web Service Offerings Infrastructure
WSOL	Web Service Offerings Language
XML	eXtensible Markup Language
ETTK	Emerging Technologies Tool Kit (IBM)
RSA	Rivest Shamir Adleman
R&D	Research and Development

Chapter 1

Introduction

1.1 Motivation

Enterprise Application Integration (EAI) has been a driving force behind application and information system in the last few years. It has led to unrestricted sharing of data and business processes among connected enterprise applications within enterprises. This has made today's enterprises almost universally dependent on emerging integration solution such as Web Services to conduct business and deliver vital services. Likewise, means to ensure regular operation of enterprise applications a mainstream requirement.

In chemical corporations such as Bayer AG, the IT department has a multitude of applications at its disposal, to support its laboratory researchers and services, it provides worldwide. Such enterprises are currently using the Web Service technology as a means to interconnect desperate applications, and simplify interoperability between heterogeneous and autonomous systems for both internal and external integration. These Web Services reduce the cost of integration and automation of their applications, through centralization of common secondary modules, which will later share its service with other applications.

Current increases in volume of Web Services, which have been deployed by such enterprises, make the management of Web Services applications a serious challenge. When enterprises have several Web Service applications that are used for the integration of numerous enterprise applications, they lead to a software environment whereby the applications are interdependent. This mutual dependency implies an increase in potential risk for the whole enterprise software system, if one of these Web Service application's components is modified or fails to work properly. Likewise, each application performance

depends on the combined performance of cooperating applications. Such challenges have led to a growing need for enterprises to confront Web Service monitoring and management issues as a priority.

The Web Service technology is based on the following protocols: SOAP (Simple Object Access Protocol) that is used for exchanging information, WSDL (Web Service Description Language) that is used for describing services, and UDDI (Universal Description Discovery, and Integration) that is used for the registration of services, and enables service discovery and interaction on the internet. These basic Web Service protocols do not provide adequate support to Web Service management, as well as the management of composite Web Services. Motivated by this inadequacy, "Web Service Management" has developed as a research area in the recent years. It has been mostly addressed by some academic institutions, as well as major Information Technology (IT) community such as Organization for the Advancement of Structural Information Standard (OASIS), and IT Companies such as Hewlett Packard (HP), Computer Associate (CA), and International Business Machines (IBM) that need new solution in order to improve on the efficiency of their applications. These organizations have put forward a number of protocols that includes Web Service Distribution Management (WSDM) [1], Web Service Manageability, Web Service Level Agreement (WSLA) protocol [2], and the Web Service Offering Language (WSOI) protocol that was presented by Vladimir Tosic in his PhD thesis[3].

Several of such protocols do address some specific aspects of Web Service management such as differentiation of Service Offering [3, 4], the use of Web Service technology as a communication and integration protocol for distributed system management, and the management Web Service itself [1]. However, no related work addresses the integration of management issues such as license Management, configuration of Web Service, usage analysis (billing), secure monitoring, and management of Web Services during runtime, as addressed in this thesis.

In this thesis, we argue that for an appropriate monitoring and management of Web Services, a Web Service Management infrastructure should provide visibility into the

runtime environment of Web Services. The infrastructure should enable enterprises to securely monitor and manage the availability, accessibility, and performance of Web Services that are deployed across several remote application servers. Furthermore, it should provide support for license management, configuration management, and usage analysis (billing) among Web Services.

1.2 Why do we need to monitor and manage Web Services?

In the last few years, enterprises have adopted the Web Service technology as the main choice of Enterprise Application integration (EAI) solution. Web Service provides a means for enterprises to reduce the cost of integration of their applications and improve operational efficiency. The nature of Web Services lead to a software environment whereby the applications are interdependent, and any modification or addition of components can lead to poor performance or failure in other apparently unrelated applications. Likewise, the performance of a single application in such an environment depends on the general performance of other cooperating applications. For these reasons, a means to monitor and manage these Web Services is relevant for early detection of poor performance or failure in each given Web Service.

In order to achieve effective monitoring and management of Web Services, it is important to gain visibility into Web Service runtime activity including access to necessary information about the Quality of Service (QoS) of the Web Service. The QoS for Web Service mainly refers to the performance (e.g. response time), reliability, accuracy, availability, interoperability, and security of the Web Service. From the enterprise business perspective, the QoS information is necessary to determine if the service health quality are within the range of Service Level Agreement (SLA) made with their clients. On the other hand, access to QoS information can be directly used to resolve failures and performance bottlenecks quickly.

Furthermore, in this thesis we consider that monitoring and management of Web Service should provide support to determine the management activity such as license management, configuration management, and usage analysis (billing) within the Web Service runtime environment. It should provide capabilities to monitor the license of Web Service consumer applications, as well as enable Web Service operators to determine which consumers are having valid, corrupt, and invalid or expire licenses, as they try to access their services. It should also provide access to Web Service usage information, which will enable the Web Service operator to bill the owner of the Web Service client application. In addition, it should provide access to Web Service configuration information that enables the Web Service operator to debug the cause of a Web Service failure faster.

Therefore, it is clear that deploying interconnected applications without any means to monitor or manage them will be a disaster. Since much time and money will be lost to determine the cause of failure, if the enterprise Web Service infrastructure stop operating properly. We need a standard way to enable Web Service applications to be manageable, as well as provide a unified management environment, which allows access to management information from Web Services that are deployed across several remote application servers.

1.3 Web Service Monitoring and Management Challenges

Although existing application servers that host Web Services are integrate with specific management tools for monitoring of service performance (response time), and load distribution management within a specific commercial provider's domain. Unfortunately, they do not fully support management activity as they have limited interfaces to provide visibility into a service health quality and accessibility. Furthermore, they lack the capabilities to provided information necessary to determine the activities around the Web

Service platform. For example, current administration tools that are integrate within Web Service application servers cannot account on how many instances of a service are running on it, who is currently using a certain function of a Web Service or whether a service is running adequately in accordance with its requirements, and when a service has stop functioning [5].

When the number of Web Services deployed by an enterprise begins to grow, new challenges emerge such as managing the dependence among Web Services. Since in an environment were application are interdependent, it is difficult to access the impact that might be caused by a modification, when compare to monolithic applications (or tightly couple client/server application). This is because Web Services are loosely coupled, and it is difficult to know which application is actually using the Web Service. Current management solutions do not provide a means to understand such dependency as well as a means to troubleshoot the main cause of poor performance easily in a Web Service that is cause by another Web Service. Therefore, it is important to look for new standards and ways to monitor and manage all the Web Services that are interconnected.

Web Service providers constantly need to upgrade or fixed bugs in their Web Service application, that is, how to carry out Versioning of Web Services. Many enterprises are understandably reluctant to modify or replace any of their Web Service applications. The dilemma and major problem here is how to transit from the old version to the new version without jeopardizing operations on the Web Service consumer applications.

One of the main goals behind the use of Web Service technology is to enable Business-to-Business communication among enterprises. The major challenge that emerges is how to manage different levels of Quality of Service (QoS) across service. In a multi-enterprise environment, the QoS of one Web Service depends on the QoS of a different Web Service in another enterprise. It is very difficult for another enterprise to manage Web Service outside of their environment, as well as determine the cause of IT problems.

Likewise, it is difficult to maintain, and enforce Service Level Agreements in a Multi-enterprise environment. Considering a situation whereby a Web Service provider in an

enterprise and a Web Service consumer in another enterprise sign a service level contract on the Quality of service, while the services of the Web Service providers depends on the Services it consume from other Web Service providers. The main challenge here is how the service provider can maintain a service level in a situation where the QoS of his services depend on other resources that he does not have full control. Another challenge here is how either parties or a third party supervises the service level agreement.

1.4 Research Goals

The primary goal behind this thesis is to address the challenges face by most enterprises that have decided to adopt the Web Service technology. In this thesis, I intend to provide a suitable infrastructure for secure monitoring and management of Web Services in an enterprise environment.

To achieve this research goal, I decided to identify the requirements to completely managed Web Service within a business unit, from the Web Service operator or administrator view. I also decided to design and implement a management infrastructure for Web Services that provides the following features:

- A management application that provides a unified monitoring and management environment for numerous Web Services, and their activities deployed within business units (i.e. enterprise or department).
- Monitoring of Quality of Service (QoS):- capabilities that monitors the performance, reliability, accuracy, availability, interoperability, and security of the Web Service.
- Usage analysis features that allow the administrator to easily identify the Web Service clients of a given Web Services, and track the service they have requested.

- License analysis features that enable the administrator to access license information used by the Web Service clients to accesses a particular Web Service.
- Web Service configuration feature that allows the administrator monitor and modify Web Service configuration.

In additions, another goal is to provide a framework, which enhances the development of Web Service with management capabilities, and allows other management system to monitor and manage Web Services.

1.5 Dissertation Outline

The remainder of this dissertation is organized as follows:

Chapter 2 presents an overview in to the background information related to this dissertation research. An inside in to Enterprise Application Integration (EAI) is presented and the challenges faced with current EAI solutions such as Web Service are illustrated. It further presents Quality of Service (QoS) concepts concerning Web Services.

Chapter 3 delves on the requirement analyses for Web Service management. This chapter presents an overview of SMaWS, and focuses on the investigation of the problem, and requirement for monitoring and management of Web Services within an enterprise environment using Use Case modelling technique.

Chapter 4 presents the SMaWS monitoring infrastructure for Web Service management. This chapter presents some of the fundamental reason behind the design of the SMaWS as well as conceptual solutions that fulfils the requirements mention in chapter 3. In addition, both the SMaWS architectural model and SMaWS security architecture are described

Chapter 5 describes the implementation of the various components of the SMaWS infrastructure are presented. It illustrates the feasibility and usefulness of using SMaWS framework for the monitoring and management of Web Services.

Chapter 6 presents the results of experimental evaluation of SMaWS. It describes the test-bed environment, the approach use to measure the overhead that is caused by SMaWS framework.

Chapter 7 presents an overview on some recent papers, products, and standardization works in the domain of Web Service management.

Chapter 8 concludes this dissertation by reviewing the advantage and disadvantage of SMaWS, the contributions and possible future work.

Chapter 2

Background Information

2.1 What is Enterprise Application Integration?

Enterprise Application Integration (EAI)[6, 7] is the use of software and Computer system architecture principle to integrate a set of enterprise applications in other to enable sharing of data and business process within an organization. In today's enterprise, an example of such enterprise applications are Database Management System, Supply Chain Management (SCM), Customer Relationship Management (CRM), and Enterprise Relation Management (ERM), which are often proprietary applications and designed initially to run independently without any interaction with other applications.

In the past integration is based on hand crafted codes, and several technologies such as File Transfer Protocol (FTP), Distributed Component Object Model (D/COM), Common Object Request Broker (CORBA), and Message Oriented Middleware (MOM) [64], were attempts used to enable interoperability between several distributed systems services. Nevertheless, they failed to be interoperable across different platforms. In such an enterprise environment were the integration process is not standardized, making changes can become very challenging over time as the number of point-to-point application integrations increases, as shown figure 2:1. A change to application's representation of data implies making changes to all applications that need to share that data. There is therefore a restriction of adapting business process to changes and new business opportunities.

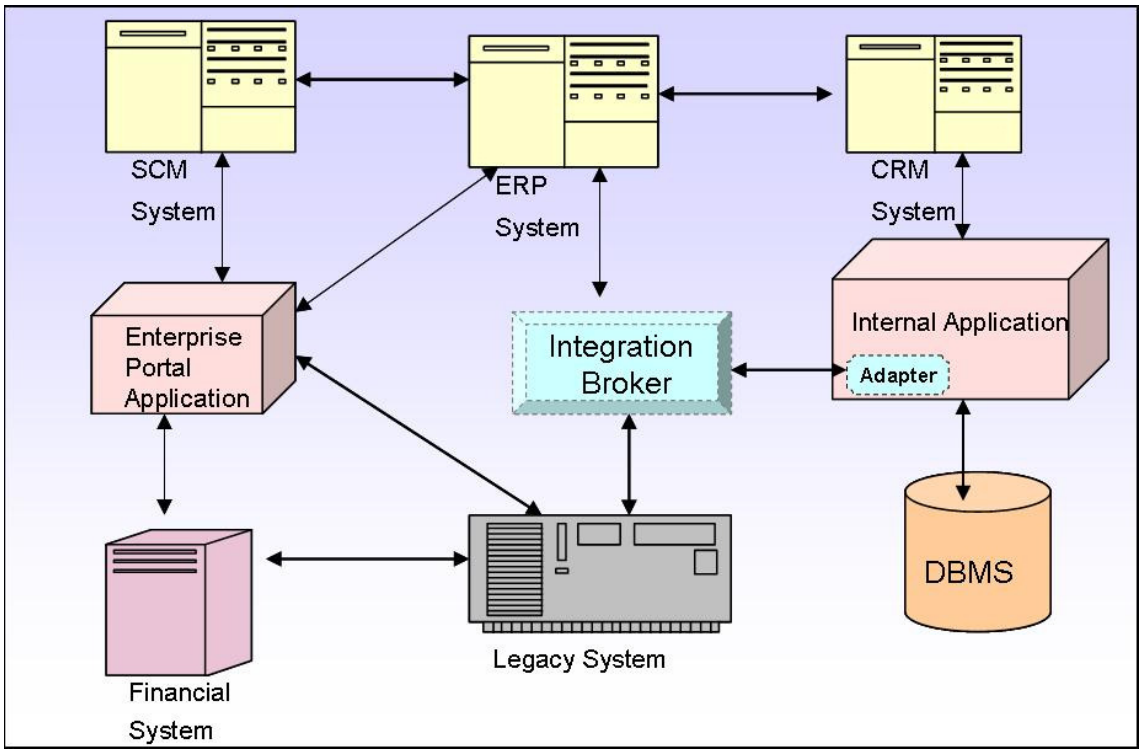

Figure 2:1 Point-to-point Enterprise Application Integration.

EAI technologies goals are to enable enterprise applications to have the ability to be interconnected, to have real time information access among them, and to streamline business processes, which consequently improve enterprise efficiency.

Identifying the need for EAI is easy, but actually accomplishing it is not easy. The major challenge in Application integration is diversity. Enterprise applications today vary widely in domains, architectures, and technologies. Any proper integration must allow the co-existence of different architectures, and solve problems surrounding integrating several technologies.

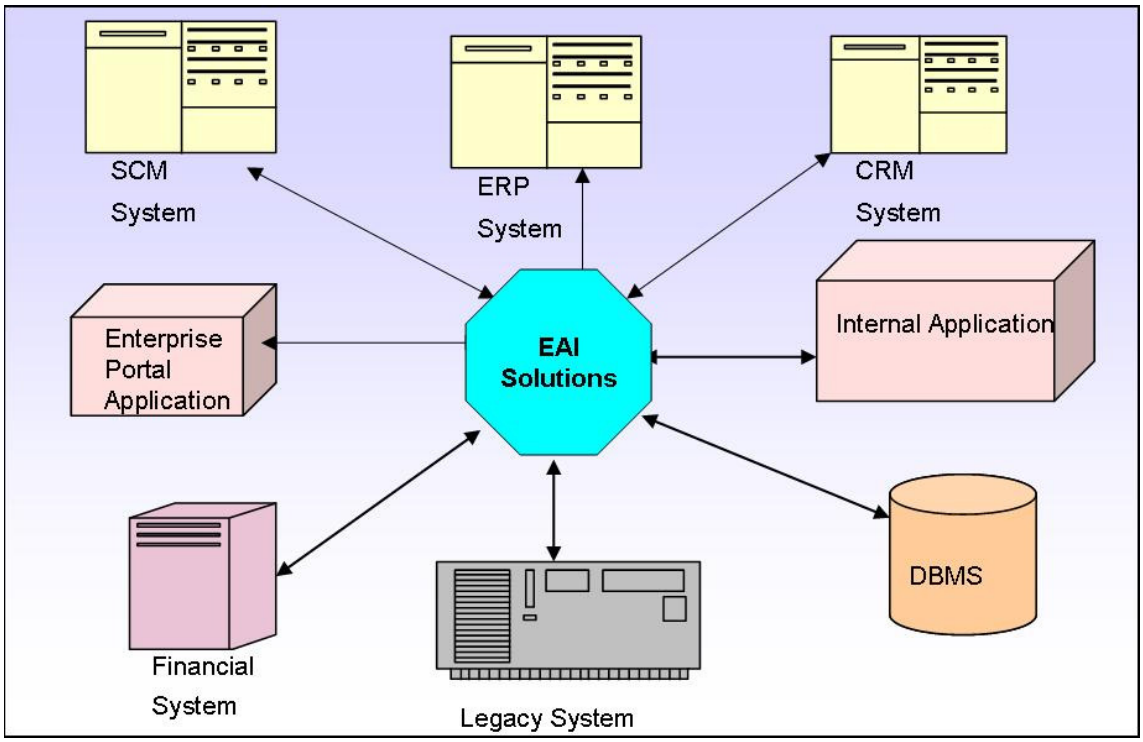

Figure 2:2 Enterprise Application Integration Solution

EAI solutions as shown in figure 2:2 represent the goals of EAI technologies to decrease the complexity and challenges surrounding point-to-point integration. It can take on many forms and exist at many levels. The appropriate level of EAI can be dependent on many factors including enterprise size, project complexity, and budget.

There are four types of integration methods to EAI:

- User Interface (UI) level Integration that occurs at the User interface, such as to enable application use a common web based Browser.
- Data level Integration that occurs at the data source level within organization and involves for example the extraction and transformation of data from application database to a common data warehouse.
- Business Process level Integration occurs at the business processing level that extend across multiple applications, and it involves the integration of

business logic to enforce the required business rules, processes, and security for the underling data.

- Method level Integration involves the aggregation common operations on multiple applications into a single application that act as a wrapper to the integrated applications. It depends on RPC (remote procedure call), distributed components (CORBA, RMI, etc), and Web Service technology (see section 2.2) to provide Application-to-Application (A2A) integration across the network.

Evolution of integration shows that Historical integration had the same challenges we have today: Performance, Availability, Reliability, Interoperability, and Security. EAI technologies are still being developed, and there is still not a consensus on the ideal approach or correct group of technologies an enterprise should use.

2.2 Web Services

2.2.1 What is a Web Service?

Web Service represents the result of the effort to address the challenge to Enterprise application Integration (EAI) or automated business integration, which requires interoperability between a variety of IT resources and applications in an enterprise heterogeneous IT environment. Web Services paradigm has emerged as a powerful mechanism for integration, as it combined the best aspects of both component-based development and the Web.

Services offered by Web Service are network-enabled components, and tend to have more in common with component-based services, since at the higher layer they all provide functions that can easily be reused without knowledge on details of how they are implemented. The Web Service uses web-based protocol (SOAP /HTTP) instead of component-based that tends to depend on the use of pipes as its communication channel.

Web Services are completely independent across vendors, platforms, and language implementation, which make them ideal for integration of enterprise applications.

There are numerous definitions given to Web Service, ranging from the highly technical ones to the more simplistic. For example, the World Wide Web Consortium (W3C) organization [65], which establishes the standards for Web Services, defines it as follows: "A Web Service is a software system identified by a URI (Uniform Resource Identifier) whose public interfaces and bindings are defined and described using XML." Its definition can be discovered by other software systems. These systems may then interact with the Web Service in a manner prescribed by its definition, using XML-based messages conveyed by Internet protocols." A simpler definition is, "a Web Service is a software application that is accessible on the Web through a URL (Uniform Resource Locator), which is accessed by clients using XML-based protocols, such as Simple Object Access Protocol (SOAP). The SOAP uses a transport protocol such as HTTP to carry the SOAP messages back and forth. Clients access a Web Service application through its interfaces and bindings, which are defined using XML notion, such as a Web Service Definition Language (WSDL) file."

Web Services are language and platform independent. A call application (Client) of a Web Service can be written in any language, and run on any platform. In other words, a client application written in Java and running on Windows could call a Web Service application written in C++ and running on Linux.

2.2.2 Service Oriented Architectures and Web Services

The Web Service technology is a connection technology in a **Service-Oriented Architecture (SOA)**, and represents the most developed example of SOA as specified by the World Wide Consortium (W3C)

SOA can be simplified as an abstract model that expresses a collection of services. A service is an implementation-independent reusable business function that is well defined,

self-contained, and do not depend on the context or state of other services. The Self-containment of services means that a service has large autonomy in decision-making, including the ability to refuse or accept request from consumers.

The SOA goals are interoperability, location transparency, and loose coupling, which are supported by the Web Service technology. An applications designed using SOA provide the same functionality as found in a monolithic architecture, and they are coupled with the following additional benefits:

- Easier extension of legacy logic to work with new business functionality as business needs increases.
- Greater flexibility to change without the need to constantly re-architect for growth
- Cost savings by providing straight-forward integration

2.2.3 Web Service Architecture

The Web Service architecture [8, 9] contains three main components: Web Service provider, Web Service consumer, and the Web Service registry. In addition, it has three main operations: publish, find, and binding as shown in figure 2:3.

A Web service provider is the application implementing the Web service. It implements the business logic, and exposes this business logic through well-defined interfaces. The Service provider is in charge of the administration of services within its domain. For example setting policies on what consumers can access, and the service at what service costs. Once a service provider creates a Web Service and its service definition, he then publishes the service within a service registry based on a standard called the Universal Description, Discovery, and Integration (UDDI).

A service registry (UDDI) is a repository used by Web Service providers to publish their Web services, and used by Web service consumers to dynamically discover, and invoke Web services The UDDI registry provides the Web Service consumer with a WSDL service description, and a URL (uniform resource locator) pointing to the service itself. UDDI registries can be a private UDDI Registry or a global UDDI Business Registry. Private UDDI Registries are used for publishing services that are only accessible to other departments within the enterprise. Whereas Global UDDI Business Registry is a global online directory for publishing global services, and gives business a unique way for companies to understand the methods necessary to conduct electronic-business (e-business) with a particular company.

Web Service consumer application can be any application that locates a Web service, and invokes the operation it provides. It finds services using the registries, and accesses these services directly through the exposed UDDI interfaces. The Web Service Consumer uses the information, received from the Web Service registry, to directly bind and invoke services on the Web Service provided by the Web Service provider.

The following figure shows how a service consumer can also be a service provider. It shows how Web service provider and the Web service consumer application are loosely connected to each other. The Web Service provider application needs to publish the service in registry to enable the Web service consumer application to find and locate them. The connection between the Web service provider application and the Web service consumer application involves data and messages that are as XML over HTTP.

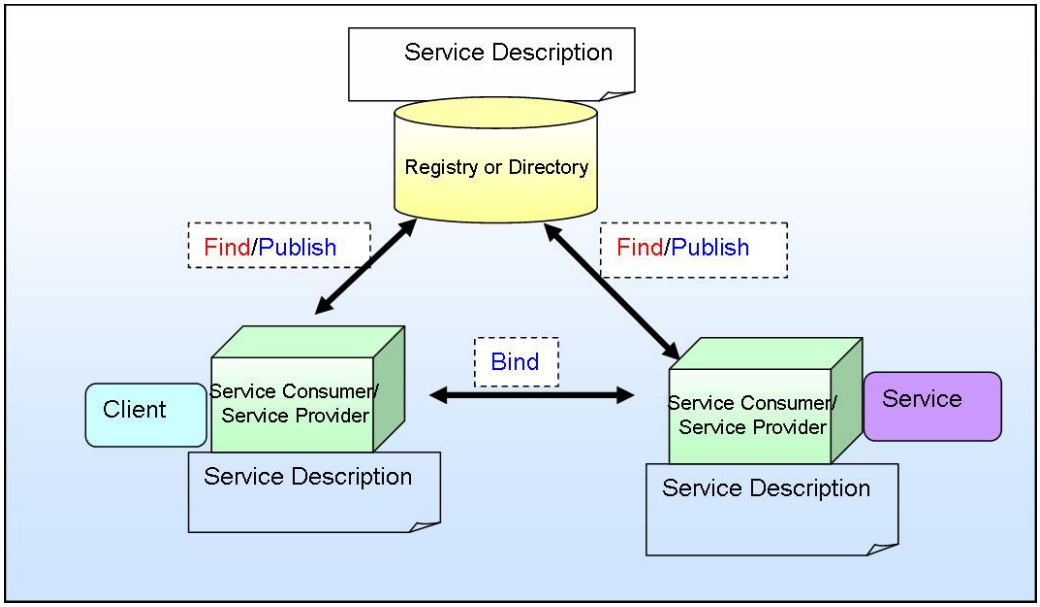

Figure 2:3 Web Service Architecture

The technologies used by the Web Services stack to achieve interoperability are shown in figure 2:4.

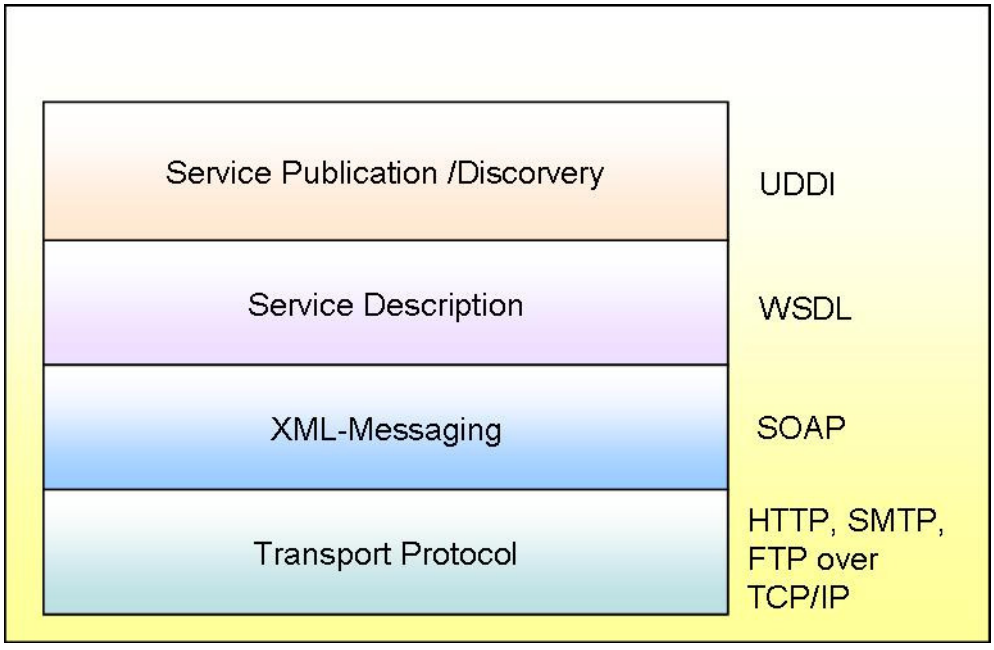

Figure 2:4 Web Service Stack (Inspired by [10])

Universal Description, Discovery, and Integration (UDDI)

The Universal Description Discovery and Integration (UDDI) [10] provides the registry piece in the Web Service architecture. UDDI defines the registry in which available Web Services are stored, indexed, and organized for discovery by the Web Service Consumer. It provides a simple framework for describing any kind of Web Services. The schema defines four core types of information: business information, service information, binding information, and information about specifications for services.

Web Services Description Language (WSDL)

WSDL is defined in a joint effort by Ariba, IBM and Microsoft [10, 11, 67]. WSDL is an XML file used to describe Web Services definition and signature for each method. It describes the purpose of the Web Service, that is, what it does; the methods that can be invoked; the parameters that need to be passed to the methods; the binding protocols used; where to locate the Web Service (URL); and where to generate remote interfaces.

Simple Object Access Protocol (SOAP)

SOAP [66] is a lightweight XML-based protocol that supports RPC (Remote Procedure Calls) and Messaging over any network protocol but primary over HTTP. SOAP is the standard mechanism for invoking Web Services, and it provides a way to communicate between applications running on different operating systems, programming languages, component models, since it uses an XML-based protocol instead of a binary format.

Transport Network

The transport network represents the standard internet protocol such as TCP, HTTP, and SMTP, which may be use to carry the SOAP protocol to enable interaction between Web Services and Web Service consumer application.

2.3 EAI and Web Services

Web Services are just another technology that enables EAI (Enterprise Application Integration), despite the fact that Web Services can be used for providing services to human end-users; their primary goal is to provide solution or address the difficulties found in EAI. Web Service enables application-to-application (A2A) integration, as well as in business-to-business (B2B) Integration between enterprises and their partners. Although there exist many EAI solutions, the reasons while Web Service EAI solutions is prefer over other traditional EAI solutions such as MOM (Message Oriented Middleware), DCOM, and CORBA are:

- Simplicity: – Web Service are easier to design, develop, and maintain, since once a framework for developing and using Web Services is developed, it is easier to automate new business process that aggregate common functions or operations spanning across multiple applications.
- Open Standards: Unlike proprietary EAI solutions, Web Services are based on open standards such as UDDI, SOAP, HTTP and this is probably the single most important factor that would lead to the wide adoption of Web Services. The fact that they are built on existing and ubiquitous protocols eliminates the need for companies to invest in supporting new network protocols [7].
- Flexibility: Web Services are loosely bound collections of services in their nature. It is easier to make changes within the Web Service infrastructure, compare to traditional EAI solutions that are very rigid, and much static in nature.
- Cheap: Web Services are cheaper, faster to implement and maintain as compare to other EAI solutions, such as message brokers that are very expensive to implement.
- Efficient: Web Services allow applications to be broken down into smaller logical components, which make the integration of applications easier, as it

is done on a granular basis. This makes Web Services solutions for EAI much more efficient than traditional EAI solutions.

2.4 The Use of Web Service within Enterprises

2.4.1 Company Overview

Bayer AG is a multinational, pharmaceutical, manufacturer, and service provider with the headquartered in Germany. It has operations and development facilities in most countries worldwide. Its operational business is focus on health, agriculture, polymers (plastics, synthetic rubbers), and chemical. It is organised into a group of seven limited companies known as Bayer Chemicals GmbH, Bayer Crop Science GmbH, Bayer HealthCare GmbH, Bayer Polymers GmbH, Bayer Business Services GmbH, Bayer Industry Services GmbH and Bayer Technology services GmbH, each controlled by Bayer AG.

The Bayer Business Services offers a wide range of services under the following business units: Business Enabling & Consulting, Human Resources Services, Logistic Solutions, Science & Technology, Procurement, Laws & Patent, Finance & Accounting, Training & Development, and IT Operation. In other words, these sub groups are responsible to provide common services such as human resources services, logistic solutions, Financial Services, and laboratory IT solutions to the other above mention companies.

The Science & Technology business unit where this PhD was carryout offers a series of laboratory Information Management system (LIMS) to support the company laboratory researcher worldwide. LIMS are applications used in the laboratory for the management of samples, laboratory staff, and instruments, as well as perform functions such as inventory, invoicing, plate management, and work flow automation. The department has developed LIMS, which are based on web-based application that are access via the intranet. These web-based LIMS are known as Integrated Chemical Information System

(ICS). Some of the ICS Web applications are ICS Beilstien, ICS Bulk Chemicals, ICS CodeInfo, ICS Image Online, ICS Naming, ICS Patent Plus, ICS Reaction Web, ICS Research Report online, ICS View, and ICS VCplus.

In a pharmaceutical company such as Bayer, a large number of compartmentalisation continues to exist, even where scientists from different disciplines are working together on the same drug projects. In the past, the process of streamline their business process was difficult as department manager had the power to make their own choice of LIMS application, and to customize the LIMS application to meet the needs of a single laboratory or R&D site. Little, if any, attention was paid to communication or standardization with other laboratories or departments. In an effort to integrate pharmaceutical research data into one global hub, IT staff at Bayer had found it difficult to combine the private data collections of chemists, biologists, and other scientists because they were 'all more or less isolated from each other, and used different definitions, and software. Information was routinely swapped between departments by staff manually by e-mailing Microsoft Excel files to each other, and at the end of a project, data was often completely lost or error prone. This approach has meant that the company have had to bear the costs of managing multiple systems, as well as work with corrupt files. Likewise, there was a potential risk for researcher to work with result data from separate research sites that was intentionally manipulated. On the other hand, sequential mergers, especially within the pharmaceutical sector have aggravated these problems of integration.

EAI solution such as Web Service has enabled pharmaceutical company to address these above mention challenges.

2.4.2 An example of Web Service Infrastructure within pharmaceutical company

Figure 2.5 shows an example of how Web Services might be use within such pharmaceutical industry. In this example, sample of theses Web Services are

Authentication Web Service, ERP Web Service, SCM Web Service, CRM Web Service that act as wrapper to around LDAP Server, ERP Server, SCM Server, CRM Server respectively. Other Web Service such as Logbook Web Service, Structure Web Service, and Naming Web Service are connected to the oracle database, which provide access to specific databases use in their business logic.

The main reason for having these Web Services is due to the sharp increases in enterprise applications, and high costs involved in the implementation of modules, such as for authentication, authorization, access to customer data, access to the human resource data, and update of customer properties in each new application developed. Further more, these Web Services allow the real time sharing of data, and business process among different applications that are dependent on them.

Remember that this example is for illustration only, and it is not intended to show a specific architecture of enterprise applications infrastructure within Bayer.

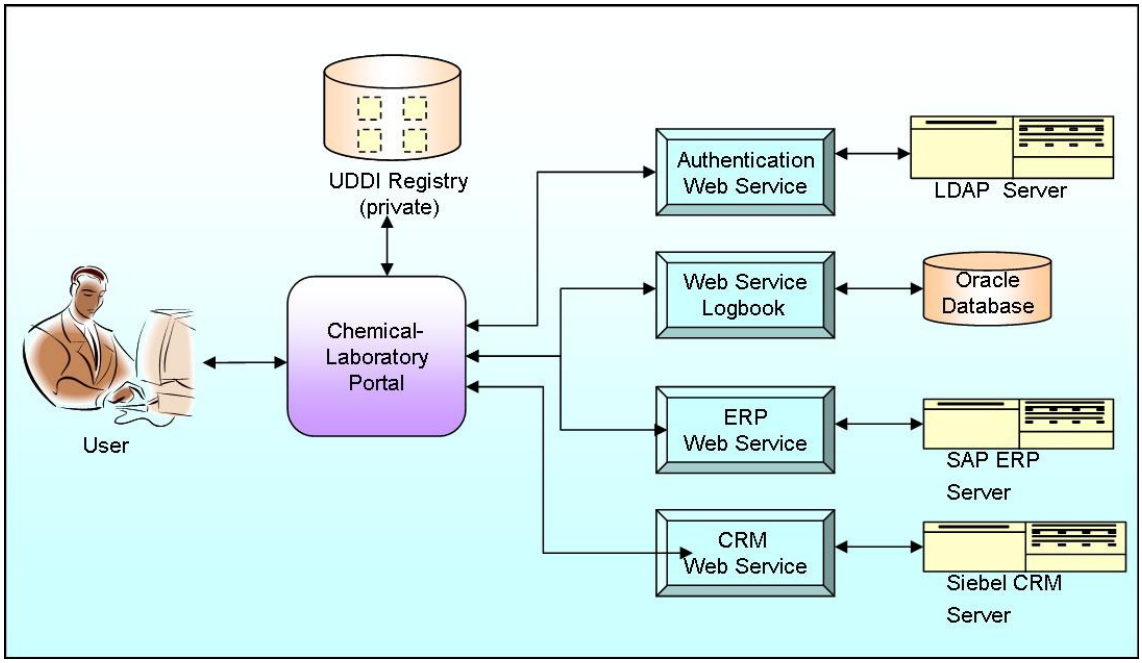

Figure 2:5 Example of a Web Service's infrastructure within a pharmaceutical company

The diagram shows how a chemical-laboratory portal application, which aggregates information from multiple internal Web Service applications, provides a single point of entry into business spread across those Web Services. The portal is loosely integrated with the Web Services via the private UDDI registry that is use by the deployed Web Service to publish the Web Service information. The portal later uses the private UDDI registry to find information about these Web Services, and invokes their services over the intranet. The Web Service binding information (Address) can be hard coded in the Web Service consumer application to avoid resource intensive, and time consuming in finding the Web Service binding information. The communication between the portal application, and the UDDI, as well as the communication between the portal application, and each of the Web Service is based on SOAP.

The Laboratory Information System (LIS) portal uses Authentication Web Service to authenticate user that try to logging. After the user is successfully login, the following actions do take place:

- The portal application can invoke the Authorization Web Service to determine the role of the user, and the virtual laboratory he could access.
- The portal application can invokes the structure Web Service and retrieve information about the chemical structure of the chemical it intends to order.
- The portal application invokes the CRM Web Service, and retrieves the user personal information, such as name, mailing address, social security number, telephone number, and email.
- The application invokes the ERP Web Service, and retrieves the account information, such as account number, balance and transaction history, of the user.
- The portal application can invoke the logbook Web Service to log users' activities, for usage analysis and billing.

2.4.3 Challenges encounter with the new integration solution – Web Service

In section 2.4.2, we see that while the Web Service based integration helps to streamline enterprises business processes, and consequently improves enterprise efficiency. On the other hand, the Web Service Infrastructure leads to a software environment were applications are interdependent. For example, the Chemical Laboratory portal depends on the Authentication Web Service in order to authenticate it users, as well as the other Web Services depend on the Web Service logbook to log user activities that is later used to bill the Web Service consumer. This new challenge in the form of dependency is a potential risk for the whole enterprise applications infrastructure to fail, if one of the Web Service crashes. Such challenges have led to a growing need within enterprises for means to monitor and manage Web Services as vital, in order to ensure the smooth operation of these Web Service applications.

Existing application management solutions turn to be ineffective in monitoring and management of Web Services. This is because, firstly, most Web Services are developed without the developer taking into consideration of management issues. Secondly, existing management tools that are integrated in the Web Service runtime environment have limited interfaces to provide visibility into the service health quality [5]. Thirdly, according to a Gartner study, most enterprises will use at least 25% more time than needed in troubleshooting enterprise applications, as result of failure to use appropriate monitoring and testing tools [12]. Finally, while most Web Service applications are currently built on Java 2 Enterprise Edition (J2EE) platforms, and new technologies such as J2EE Management JSR 77 [13] and Java Management Extensions (JMX) [14], which provides a vendor-neutral way of managing and monitoring resources in J2EE servers. In general, tools have not taken advantage of these technologies to make Web Services easy to manage and monitor.

Enterprise needs a suitable monitoring and management infrastructure for Web Services which enhances the existing infrastructures for hosting of Web Services. A management

infrastructure that provides details information about it activities during run-time, at the same time enables improve manageability to the existing Web Services [5]. To serve this need we propose an infrastructure called SMaWS (Secure Management of Web Services), which provides an enterprise with a Web-based management application that allow them to monitor and manage the execution of their Web Services based on SMaWS framework. The SMaWS framework is a JMX-based framework that enables Web Services to be developed with manageability capabilities.

2.5 Web Service Quality of Service (QoS)

2.5.1 QoS Overview

The definition of the term Quality of Service (QoS) is broad, as it is often refer to as the ability to reserve resources, as well as a measure of performance of a service. In the field of telephony according to the ITU standard X.902, it is define as "A set of quality requirements on the collective behaviour of one or more objects." In the field of traffic networks the term is referred to as a resource reservation control mechanisms that ensure most important data gets through the network as quickly as possible. On the other hand, it is sometimes simply defined as a quality measure of service from the user perspective.

In the field of Web Service, QoS is referred to as a measure of different quality and properties of a Web Service that includes availability, performance, reliability, safety, and security. Web Service QoS requirements result to both quantitative aspects that can be measured as well as qualitative aspects that can not be directly measured. The quantitative aspects are availability, performance and reliability. The qualitative aspects are safety and security.

Web Service QoS is also often refers to as range of techniques that guarantee a certain level of QoS to a service requestor. The architecture of a Web Service has several factors

that influence it QoS, and this range from the configuration of the computer system hosting the web service, over to the Back-End Systems, the programming language as well as the communication protocol use.

In this thesis, one of the main goals was to develop techniques to enable the determination of the QoS of a Web Service from the service provider's perspective.

Web Service QoS aspects are define as follows:

- **Reliability**: Reliability is the probability that over a given period, the Web Service is capable of maintaining the service and service quality.
- **Performance**: Performance of a Web Service represents the number of Web Service requests served at a given time period.
- **Availability**: Availability is the probability that the Web service will be up, running, and able to deliver services at any given time.
- **Safety**: Safety of a Web Service is the judgement of how likely it is that the Web Service will cause damage to its environment.
- **Security**: Security of a Web Service is a judgement of how likely it that the Web service can resist accidental or deliberate intrusion.

In the following section, explains how SMaWS determine each of these aspects of QoS of a Web Service.

2.5.2 Monitoring the QoS for Web Services

2.5.2.1 Reliability

One of the complex aspects of Web Service QoS requirement is reliability. Reliability defines the probability that over a specified given period time, the Web Service is capable of maintaining failure- free operation in a given environment. In order to be able to estimate the reliability of a Web Service, the following three dimensions must be considered:

- *Hardware reliability:* the probability (P_H) of the hardware component failing. This includes the reliability of both hardware and software infrastructure hosting the Web Service.
- *Web Service application*: the probability (Pws) how likely the application will produce the expected output.
- *Operator reliability*: the probability (P_o) how likely the operator of the Web Service will make an error.

The reliability probability of an overall Web Service system depends on the reliability probability of the hardware, reliability probability of the Web Service application, and the reliability of the operator:

$$P_{reliability} = P_H * P_{ws} * P_O$$

Considering that Web Service applications have no influence on the reliability of the computer system hosting it, as well as the activities of the operator administering it, we assume that the hardware infrastructure is reliable as well as the operator and their reliability probability are equal to one (i.e. $P_H = 1$; $P_O = 1$). Therefore, the reliability probability of the Web Service system depends only on the reliability of the Web Service application

$$P_{reliability,ws} = P_{ws}$$

The focus of this work is not to predict the reliability of a Web Service, but instead, to determine the QoS metric that describes the activities of the Web Service. Therefore, following quantitative aspects of Web Service reliability are considered:

- *Average Response Time*: The average Web Service response time in seconds for total number of Web Service requests since start-up.
- *Success Count*: The total number of successful Web Service invocations.
- *Failure Count*: The total number of failed Web Service invocations.

- *Authentication Success Count*: The total number of successful authentication.
- *Authentication Failure Count*: The total number of times an authentication failure occur.

2.5.2.2 Performance

The performance of a Web Service refers to the quality of service aspect that is measured in terms of throughput. *Throughput* represents the number of Web Service requests processed over a specified period of time. A higher throughput represents a good performance of a Web Service.

The throughput metric is directly dependent on the response time of a Web Service processing a request. The response time is made up of the processing time (the time required by the CPU to process a request) and the delay time experienced by the request to be processed.

Predicting the performance of Software systems has been approached in the past and a comprehensive survey of modelling approaches for performance prediction is presented in [68]. These models have performance parameters such as I/O utilisation, CPU cycles or network characteristics, specified by the developers in order for the performance predictions to generate meaningful results. It has been proven that such techniques and tools like SPE-ED helps in achieving performance goals [69].

However, middleware application and other component-oriented platforms, such as Web Service exhibit an inherent complexity, which developers find hard if not impossible to quantify even in simple models.

Automated services such as caching, pooling, replication, clustering, persistence or Java Virtual Machine optimisations, provided by application servers, contribute to an improved and at the same time highly unpredictable run-time environment. Furthermore, application server implementation can vary greatly from vendor to vendor in respect to these services.

It is therefore impossible for developers to create performance models for their application and specify the mapping of methods to processes or instances to processors, I/O characteristics, or CPU utilisation.

2.5.2.3 Availability

Availability of a Web Service defines the probability that at a point-in time, the Web Service will be operational and able to deliver the requested service. Larger probability values represents the service is always present and ready for immediate use, while small values indicate that it is difficult to foretell if the service will be available at a given time. The availability of 0.99 means that in every 100 time of unit the Web Service is likely to be available for 99 of these. Associated with availability is Time- to-Repair (TTR). *TTR* represents the time it takes to repair the Web Service that has failed.

In this thesis, the availability refers to whether a Web Service is running and available for consumption. Therefore, the availability is represented by a set of Boolean values (true, false) that indicate if the current state of the Web Service is either up or down.

- *Up* indicates the Web Service is capable of accepting and processing requests (that is, service is deploy and running), and
- *Down* indicates the Web Service is not able of accepting any requests (that is, service is undeploy or stop running)

2.5.2.4 Safety

The safety of the Web Service is a qualitative aspect of a Web Service that reflects it ability to operate without causing damage to it environment. Safety measures adopted in the web service do ensure that incorrect input data used during service request do not lead to a Web Service malfunction.

SMaWS infrastructure determines the safety of the Web Service by Web Service testing. This involves a series of functional tests, which determine the reaction of the Web Service after a series of different input data are assigned.

2.5.2.5 Security

Security of a Web Service is a qualitative aspect of Web Service that reflects the degree it can assure confidentiality by authorizing the parties involved, encrypting messages, and providing access control.

In this work, I determine this QoS aspect of a Web Service by testing the security of the Web Service for potential vulnerabilities.

CHAPTER 3

Monitoring and Management of Web Services: *REQUIREMENT ANALYSIS*

3.1 Introduction

3.1.1 SMaWS Overview

As I mention in subsection 1.2, the nature of Web Service leads to a software environment whereby applications are interdependent, and the performance of a single application depends on the general performance of other applications. It is necessary to provide a means to monitor and manage Web Service, for earlier detection of poor performance or failure in each given Web Service.

This dissertation describes SMaWS infrastructure, which represent a vision about, how to enable enterprises to provide secure monitoring, management, and configuration, for their distributed Web Services within various business units (departments) remotely. The term SMaWS is an abbreviation for "Secure Management of Web Services". SMaWS framework is intended as a foundation for building enterprise based monitoring and management infrastructure for Web Services. SMaWS framework enables developers to develop Web Services with manageability capability that allows the Web Service's administrator to monitor and manage them. SMaWS fulfils the requirements mention in the next sections. These are:

- *Monitoring of Web Services:* SMaWS infrastructure must provide means for enterprise to monitor and determine the identity, performance, reliability, availability, security, and usage of a Web Service. This enables

enterprises to proactively monitor all its Web Services with pre-defined thresholds, and to identify performance degradations early enough to prevent failures.

- *Management of Web Services:* SMaWS infrastructure must provide means for enterprises to carry out management activities such as configuration, usage analysis, and analysis of license used by a given Web Service Consumers.
- *Security:* SMaWS infrastructure must provide a secure environment for monitoring and management of Web Services. It must provide means to execute security policies.

Web Services are server-side applications that run in an application server, and are completely User-Interface less, which makes it difficult to determine its availability and reliability, as compares to other applications such as Web application that have a user-interface via which unavailable service could easily be detected. The SMaWS infrastructure provides a web-based management application called SMaWS Manager that allows the user to monitor and manage remote Web Services distributed across an enterprise environment.

Figure 3:1 shows the use of SMaWS infrastructure in the management of Web Services within an enterprise environment. In this environment, the Web Service consumer interacts with Web Service Application via the private UDDI Registry. The user uses the SMaWS Manager to monitors and managed all the Web Services that are deployed and register in the SMaWS registry. The SMaWS Manager application interact with the Web Service application it intends to managed, after it had retrieved the Web Service agent address from the SMaWS registry. The SMaWS Manager provides means for automatically discovery of Web Services that are deployed.

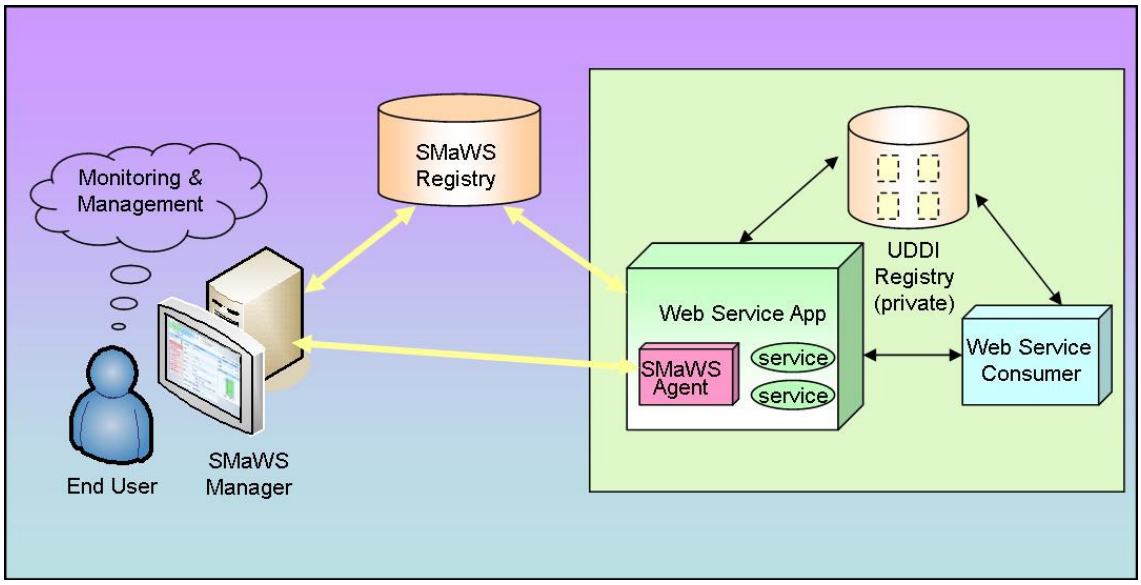

Figure 3:1 SMaWS Overview within an Enterprise environment

3.1.2 Use Case Overview

In this thesis, the Use Case modelling technique is use in other to enable understanding as well as to establish and describe the requirement of the system. Use Cases are an effective communication tool between developers and system consumer because they simply describe the business process that the system must support, without any details about the technical jargons. It is important to note that there is no real standard for use cases, so the Use cases approach that is used in this dissertation is a combination to both Use Cases based-on textual approach [15] and graphical-based UML (Unified Modelling Language) approach uses case view [16]. This combination provides textual description, and graphical representation of the functionality and behaviour of the system as perceived by external users.

Use Case is an artefact explicitly included in the requirement analysis phase of most Software engineer methodologies such as the Object Oriented approached. It defines goal-oriented sets of interactions between external actors and the required system. It defines behaviour required from the system.

Use cases are sequence of actions that an actor performs within a system to achieve a particular goal.

An ***actor*** represents the entity (either a person or another external system) that interact with the system.

- *Use case Scenarios* are instances of a Use Case and a single path through a Use case.
- *Scope* represents the extension or limitation of what is consider in the project.
- *Stakeholders* represent individuals or groups with interest in the success of the project.

However, Use cases do not offer a complete specification for our system because Use cases are usually not effective for the specification of system characteristics such as performance and availability.

3.2 Requirement analysis

3.2.1 Description

Our analysis shall focus on needs of enterprises that wish to have an effective means for monitoring and managing of their Web Services they have deployed. These enterprises need the ability to monitor and manage the performance, heath status, usage, licenses of Web Services across the enterprise, as well as to monitor the dependence among their Web Services and other enterprise applications that use these Web Services.

Finally, our focus shall also be on the need to enable remote monitoring and management of Web Services, as well as the need to provide a secure Web Service monitoring infrastructure. The Web Service Monitoring and management system should provide means that ensure unauthorized user cannot access the system. This means that, there

should also be a secure communication between the Management application and the various Web Service applications its monitors and manages.

Scope

We shall limit ourselves to the management of Web Service within an enterprise that uses Web Service as a means to achieve Enterprise Application Integration (EAI). Therefore, issues concerning the management of co-operation or contracts (Service Level Agreement [22, 42]) within composite Web Services (that is, between Web Services deploy across the internet by several Web Services provider) is not address.

Primary actors

In our Use Cases, we consider that an enterprise do have different group of users that could monitor or managed Web Services depending on their role. These groups of users are the Administrators, operators, and help desk persons.

- The *Administrator* is a self-evident actor that exists already and is not bound to the schema of the Management application. Administrators have the right to monitor and manage all the Web Services that are deployed within the enterprise. The administrator is responsible to create and assign management application's users with the role operator or helpdesk to specific Web Services.
- *Operators* have the right to monitor and manage a given or group of Web Services assign to them by the Administrator.
- *Helpdesk* users only have the right to monitor a given Web Service or group of Web Services assign to them by the administrator.

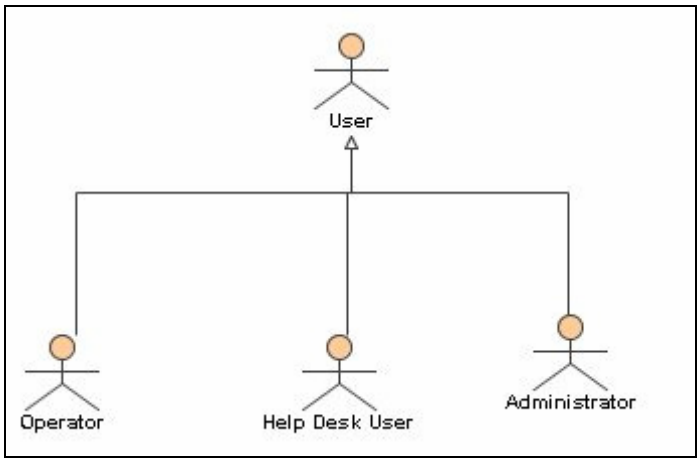

Figure 3:2 System Users

Stakeholders and Interests

- The Actor wants to monitor or manage Web Services.
- The enterprise wants to see the smooth running of it Web Services.

3.2.2 Usage Scenarios

The following Use Cases describe how a user would use the management application to monitor and manage Web Services. Associated to each of the use case scenarios is the minimal guarantee that sufficient logging information will exist, so that all activities could be detected when something goes wrong.

3.2.2.1 Use case - Global View

Context of use: The Use case global view describes the main goal in this thesis. A user uses a management application to manage and monitor a Web Service or a group of Web Services within a business unit.

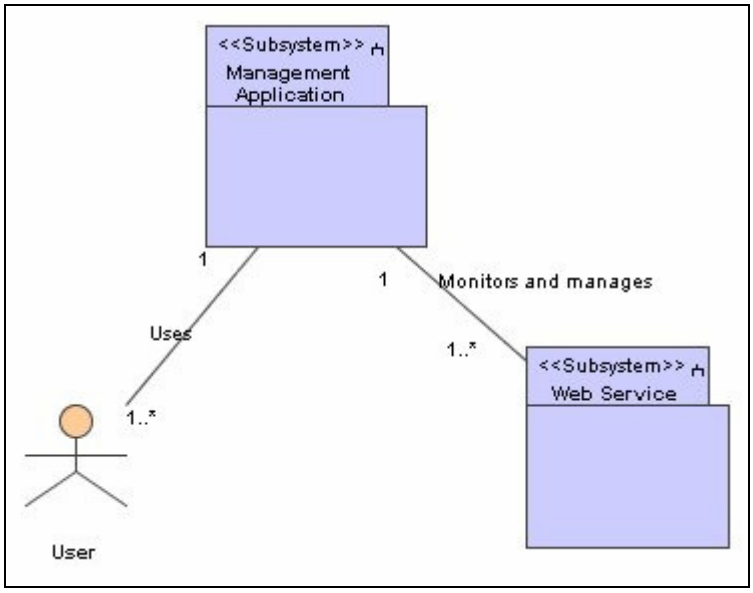

Figure 3:3 Use Case –Global View

Level: Summary

Precondition: None

Trigger: User access the management application via a web browser.

Main Success Scenario:

1. The user logs in to the management application in order to monitor and manage distributed Web Services running on variety of platforms.
2. The management application locates the deployed Web Services.
3. The management application establishes a secure communication channel with each Web Service the user is authorized to monitored or managed.
4. The Web Service provides the management application with a set of management operation that exposes a range of management capabilities.

5. The management application response to the user request by presenting the user with an interface, where by he can monitored, and managed selected Web Services depending on his role.

Requirement:

1. A mechanism is required to enable the management application to manage and monitor Web Services independent of the platform or language of implementation.
2. Description of management information needs to be established, in order to achieve real integration of management information between the management application and Web Services running on variety of platforms.
3. The system needs to enable remote monitoring and management of Web Services
4. The management application needs a mechanism that enables secure management of the Web Service.
5. The Web Service needs to be developed with management capabilities.

3.2.2.2 Use Case Monitoring of Web Services View

Context of Use: User wishes to determine availability, and reliability of their Web Services via the Management application. The management application provides a means to determine Web Service that have been deployed, as well as information about their identification data, configuration data, performance information, QoS metric data (e.g. Average response time, throughput, etc) and latest failure messages. On the other hand, it will enable the enterprise to proactively monitor all its Web Services with pre-defined thresholds to identify performance degradations early enough to prevent failures.

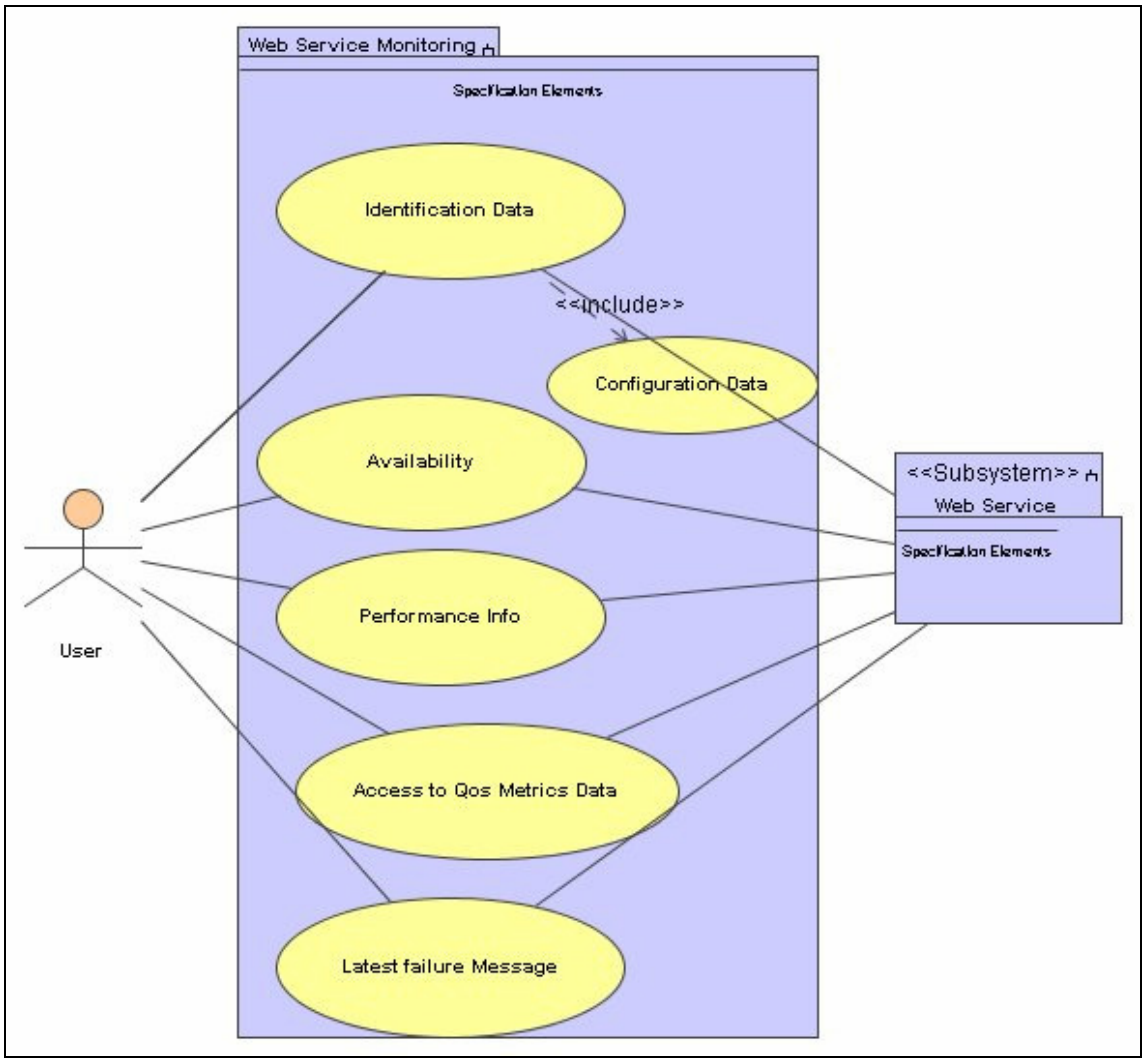

Figure 3:4 Use Case - Monitoring of Web Services

Level: User-goal

Precondition: Before this view begins, the user has logged onto the application and granted access to monitor specific Web Services.

Success Guarantees: User can access information about the Web Service identification information, QoS metrics data, Performance information since start up, and since the last ten minutes.

Trigger: User login in to the Web Service monitoring application, and has being granted access to monitor at least one Web Service.

Main Success Scenario:

1. The user sees all the lists of Web Services that he is authorized to monitor.
2. The user sees the status (deploy or undeploy) of the various Web Services.
3. The user can see the performance information of the various Web Services.
4. The user chooses one Web Service from the list of Web Services.
5. The management application presents the user information about the Web Service identification information, QoS metrics values and performance information.
6. The management application also presents the user information about the given Web Service backend resources, and the latest failure message that occur in the system
7. The user could logout.

Extension:

1. The user can click to the Management View if he has the right to manage a give Web Service.

Requirement:

1. The Web Service Identification data, Configuration data, and QoS metrics need to be determined.
2. Monitoring information about the Web Service backend resources need to be determined.
3. A mechanism is required to identify performance issues and bottlenecks via measuring actual versus expected performance.
4. The mechanism is required to allow the user establish performance thresholds.
5. The Management application should provide a means to enable the user to access the latest failure message, as well as all the failure messages collected from the exceptions raised since Web Service application was deployed.

3.2.2.3 Use Case – Management of Web Services

Context of Use: A User wishes to administrate and configure the Web Services he is authorized to monitor and manage. Management application provides users the following features: QoS management view, Web Service testing view, Web Service Configuration view, Web Service Usage analysis view, License Management view, and User Management view

.

Figure 3:5 User Case – Management of Web Services

Level: User-goal

Precondition: Before this view begins, the user has logged onto the application and granted access to monitor and manage specific Web Services.

Success Guarantees: User can access Web Service management view.

Trigger: User clicks the management view button in the monitoring view of the Web Service management application.

Main Success Scenario:

1. The user sees all the lists of Web Services that the administrator allows him to manage.
2. The user sees the status (deploy or undeploy) of the various Web Services.
3. The user chooses one Web Service from the list of Web Services
4. The management application responses to the user by presenting a QoS Management view.
5. The user can switch to Web Testing view, Web Service configuration view, Security-configuration view, License Management view, and back to Monitoring view by clicking on the respective button within the management view.
    a. **QoS Management View**:- It provides an interface with editable fields, where he could tune the QoS metrics of a given Web Service. For example, the user can set new threshold for the Web Service performance.
    b. **Web Service Testing view**:- It provides an interface that can be used by the user to invoke or run test-operations assign to a given Web Service. These test operations enable the user to determine the Web Service reliability.
    c. **Web Service Configuration View**:- Its view provides a means for the user to access Web Service Deployment Descriptor (WSDD) file belonging to the given Web Service. Access to this file can help the administrator during failure analysis, and to re- configure the Web Service properties, such as enable or disable the availability of services offered by the chosen Web Service.
    d. **Web Service Usage Analysis view**: - It allows user track the usage of a given Web Service. It provides information about the total number of request the Web Service has processed since deployed. It also provides the means for the user to request for details information about the Web Service usage activity. On the other hand, apart of providing a means to

determine information about the list of given Web Service clients that have access a specified Web Service. It also provides a means for the Web Service operator to bill the owner of the Web Service consumer application.

e. **License Management view**: - It provides a means for a user to access license information used by Web Service consumer applications to access the services of a given Web Service. The Web Service Application stores license information, when decrypting the license presented to them by consumer application that try to access their services. The user can use the view to sort and classify these licenses into a list of invalid licenses, corrupt licenses, valid licenses, and licenses that are soon going to be expired.

f. **Security management view**:- it provide definition of security policies, users and access rights, activation and deactivation of security services, and the monitoring of the proper operation of the system.

Extension:

1. QoS Management View
2. Web Service testing view
3. Web Service configuration view
4. Web Service Usage Analysis view
5. License Management View

Requirement:

1. Determinations of the Web Service properties that need to be manage.
2. The Management application should provide the ability to set and changes the types of the managed Web Service Statistics collected
3. A mechanism is required to enable the Web Service tracks the usage of a Web Service.
4. A mechanism is required to allow the management application access Web Service usage information within a Web Service.

5. Determinations of Web Service testing operations that determines the reliability of a given Web Service and do not have negative impact on the Web Service performance while running on the production server.
6. A mechanism is required in the Web Service to decrypt and log the license information of the Web Service client that tries to access the Web Services.
7. A mechanism is required in the management application to sort the Web Service client licenses provided by a given Web Service.

3.2.2.4 Use Case – System Access

Context of Use: User wishes to log in to the management application. The Management Application provides authentication and authorization. It determines the list of Web Services the user is authorized to monitor. It later requests for connection with the respective Web Services by forwarding the user credential. Each given Web Service further authenticate the user and response by establishing a connection if the authentication was successful.

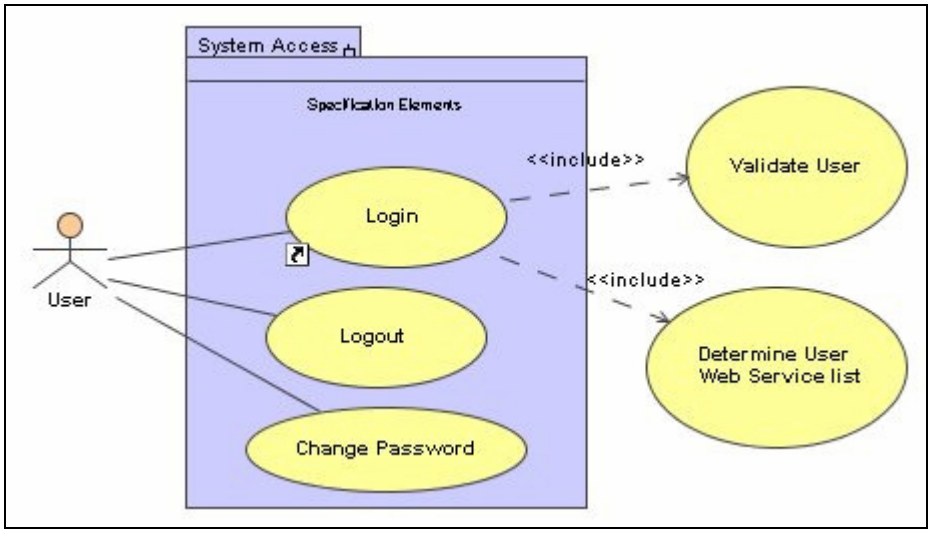

Figure 3:6 Use Case – System Access

Level: Sub function

Precondition: The Web service administrator creates an account with appropriate privileged, and issues a user a username and password. The user's workstation is a web browser configured according to his role.

Success Guarantees: User can enter to monitoring view of the management application.

Trigger: User access the manage application via a web browser.

Main Success Scenario:

1. The management application displays a page, and prompts the user to enter a user name and password.
2. The management application verifies if that user exist and the password is valid
3. The management application determines the role of the user and the list of Web Services the user is authorized to monitor or manage.
4. The management application retrieves the address of the respective Web Service from the repository that is where these Web Services publish their address for the management application to access.
5. The management application uses these addresses to requests for connection with each of the Web Services the user is authorized to monitor or manage by sending encrypted credential of the user that is successfully login.
6. Web Service agent authenticates the user credential data and determines if the user is authorized to monitor and manage the Web Service.
7. The Web Service's agents do response to the request by establishing a connection with the management application.
8. The management application then gives the user access to the monitoring view, where he can further switch to the management view depending if the user has at least a Web Service where he has the role of an operator.

Extension:

1. If the user enters a wrong username, the management application will display a message asking the user to contact the administrator.

2. If the user enters a wrong password, the management application will display a message that the password is invalid.
3. If the valid user is an administrator, the application allows the user to monitor and manage all the Web Services within a business unit.

Requirement:

1. The whole system infrastructure security risk needs to be address, that is :
    a. Management application *confidentiality*
    b. Management application *integrity*
    c. Managed Web Service *confidentiality*
    d. Managed Web Service *integrity*
2. *Secure communication* between the management application and the managed Web Services.
3. A mechanism is required to ensure that Web Services can securely send Management application their address.
4. A mechanism is required to enable the Management application ensure that Web Service's address it retrieves comes from the Web Service it intends to monitor.
5. An authentication mechanism is required to determine which user has the right to access the system, and which where they can monitor and manage. Further, the authentication should be single sign-on that is users must be able to access all the Web Service management infrastructure's resources after first successful authentication, and any restriction will depend if users have been granted authority to monitor or managed a given Web Service.
6. When a user tries to monitor or manage a given Web Service, the authorization process should check if the user has been granted permission. The authorization decision must depend on the privileges granted to the user by role to a specific Web Service.
7. Connection between the management application and the Web Service needs to be secure.

3.2.2.5 Use Case – User Management

Context of Use The Administrator wish to add new user to access the Management Application and assign a list of Web Services the user is authorized to monitor or manage. The administrator assigns the user the role operator or helpdesk to each of Web Service.

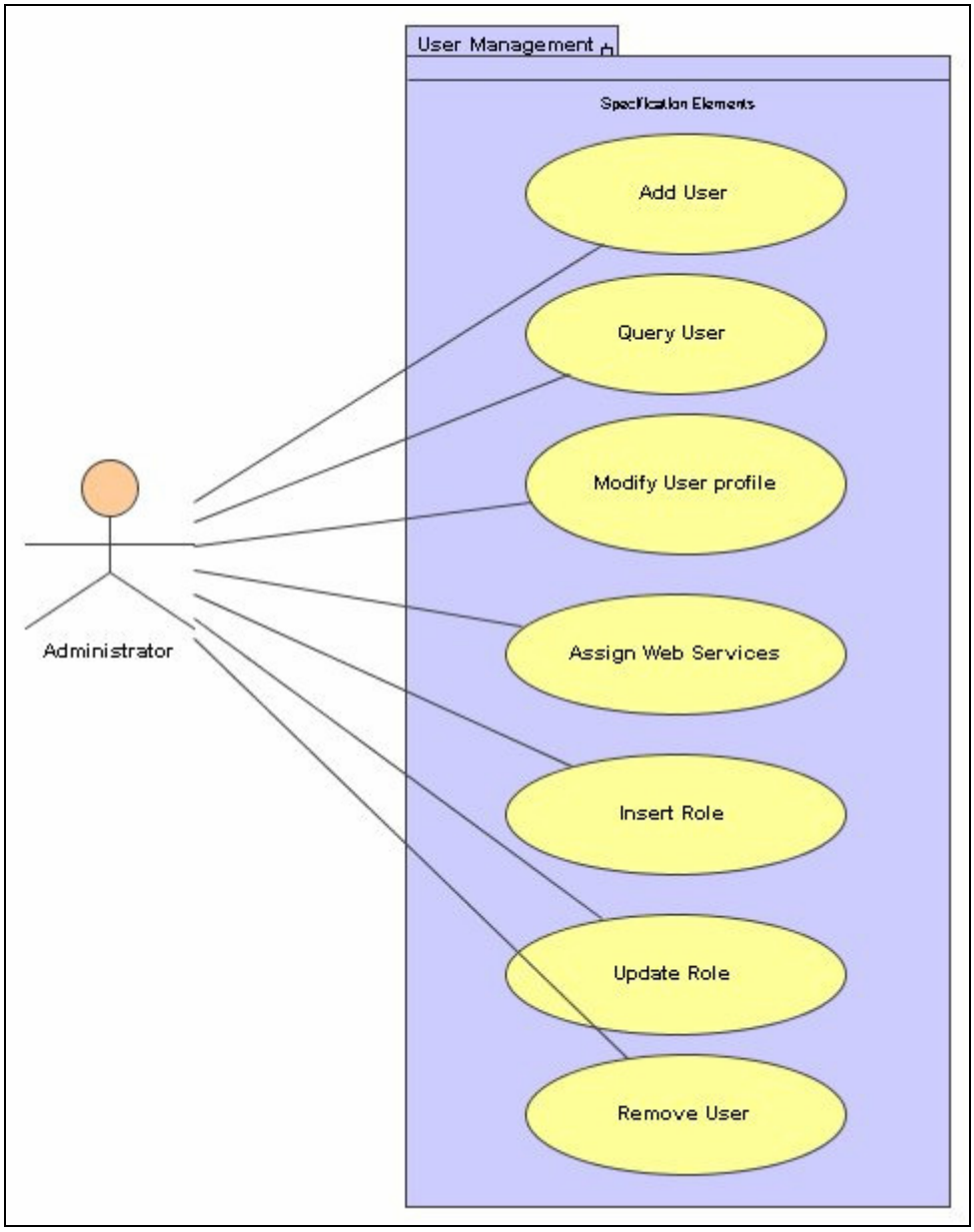

Figure 3:7 Use Case – User Management

Level: User-goal

Precondition: Before this view begins, the user has logged onto the application as an administrator and granted access to monitor and manage all Web Services.

Success Guarantees: User can access User Management view.

Trigger: The administrator clicks the User Management view button in the any of the Web Service Management view of the management application.

Main Success Scenario:

1. The management application determines if the user is log in as an administrator.
2. The management application displays the User management view
3. The administrator uses the interface to add new user and assign the user a list of Web Services with a given role.
4. The administrator can also uses the interface to query the users of a given Web Services, to modify a given user profile, to update a user Web Service role and to remove the user from the system.

Extension:

1. If the administrators try to add users that exist already in the system, the management application will display a message informing the administrator the user exist already.

Requirement:

1. A security policy is required that include the following:
    a. Determine who is allowed access to the system.
    b. Determine when they are allowed access.
    c. Determine what they are allowed to do with the system.
    d. Determine how user get added to the systems and how they get removed
2. The Management application should provide the ability insert, maintain, and remove user that could access the system.
3. The Management application should provide the ability to assign Web Services to a given user by the administrator.

4. The Management application should provide the ability to grant user privileges by role to specific Web Service.

3.3 Other Functional Goals

One set of requirements that impact the SMaWS Infrastructure are: reliability, maintainability, testability, and usability. Some others include fault tolerance, error handling, security, portability, and performance.

Existing WSDM[1] Web Service management solution lead to a situation whereby measured metrics data are transferred immediately to the management system to process and aggregate, and this result to a high use of network bandwidth. The SMaWS infrastructure is design to be fast, and to use low network bandwidth, as the Web Service agent filters and aggregate the measured metrics data in the managed Web Service Application, and forward only pertinent and aggregated data to the management application.

The SMaWS framework must be portable across a wide range of new and old Application server, since most of the existing Application server are based on Java Virtual Machine that do not have instrumentation.

CHAPTER 4

SMaWS Infrastructure

This chapter presents some of the fundamental reason behind the design of the SMaWS, as well as conceptual solutions that fulfils the requirements mention in the previous chapter. It also provides an insight into how the SMaWS framework fit in the Web Service infrastructure, and how the components within the SMaWS framework interact with the surrounding components within an enterprise Web Service infrastructure. I discuss the SMaWS architecture with diagrams that provide a close analysis of the design to ensure that the system meet all the complete requirements and is consistent in its operation.

SMaWS target java-based environment, although conceptual structure applies to other programming language such as .NET.

4.1 SMaWS Architecture

4.1.1 Web Service management approach

Web Service management approach used by SMaWS focus on realising a centralised Web Service management system that is based on the manager-agent concepts, and at the same time consume low network bandwidth. According to this concept, a single manager may control several agents. The SMaWS framework consists of a SMaWS Manager subsystem and the SMaWS agent with the managed Web Service.

The SMaWS Agent reduces the amount of messages send to the SMaWS Manager application by aggregating the measure metrics data (e.g. response time to average

response time) that is send to SMaWS Manager Application when requested. This approach enables the Web Service Management system to keep the amount of network bandwidth consume remain low.

Alternative approaches used by the Web Service Distribution Management (WSDM) also focus on realising a centralised Web Service management system, but the centralised manager application is charged with retrieving the low-level data from the managed Web Service and presenting that data in a form useful to the Web Service administrators. For a low-bandwidth environment, this approach is unsuitable because the central manager will excessively consume backbone network capacity with routine polling of traffic.

4.1.2 SMaWS Architectural Model

This section provides an overview of SMAWS architectural model with key concepts that represent logical grouping of processes (layer) involved in the SMaWS infrastructure. Each layer has a distinctive function in the architecture, and addresses separate aspects of the Web Service monitoring and management system architecture that I envisage to developed. The SMaWS architectural model provides major block that could be develop independent to enable monitoring and management of SMaWS managed Web Services.

The SMaWS architectural model consists of the following layers as shown in the figure 4:1.

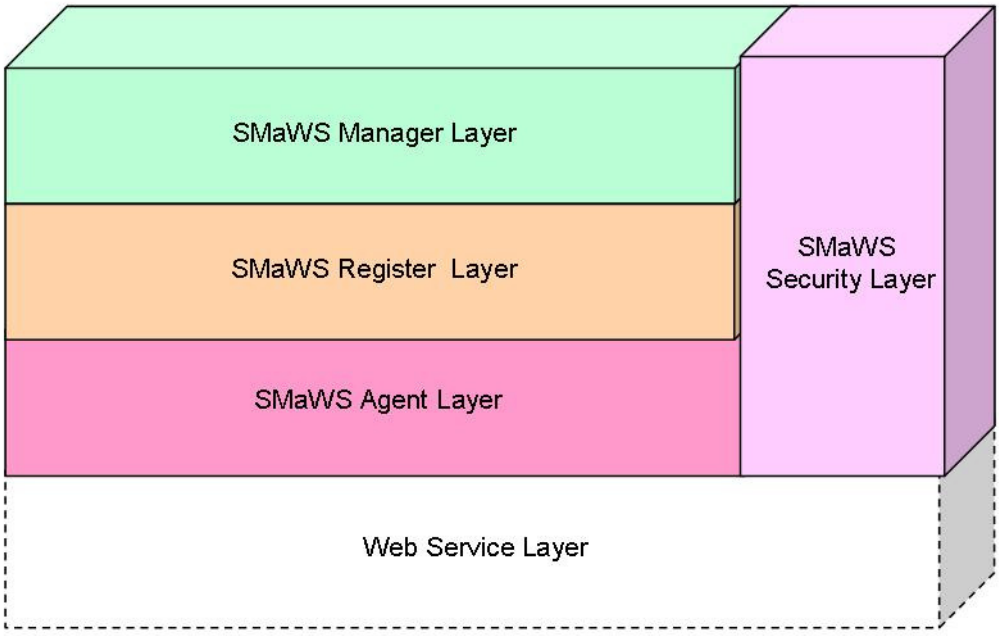

Figure 4:1 SMaWS Architectural Model

SMaWS Manager Layer is responsible to provide the user-interfaces that enable administrator to discover, monitor, manage, and configure Web Services within an enterprise. It is responsible to collect and present aggregated management data from a set of SMaWS agent in a comprehendible context to the administrator.

SMaWS Agent layer represents completely the middle layer that is responsible to enable communication between SMaWS Manager and Web Service layer. The SMaWS Agent layer is responsible to expose measurement and management data from the Web Service during runtime. It is also responsible to aggregate, and analysis the measured data against the predefined service level objects (thresholds). The SMaWS agent layer is also responsible to provide the following management activities: Web Service Configuration, Web Service usage analysis, Web Service license analysis, and Web Service testing.

SMaWS registry layer enable dynamic discovery of Web Service. It is responsible to provide a repository whereby managed Web Services can be register. It is also responsible to provide a repository whereby SMaWS Manager can search, and retrieve information about Web Services it intends to monitor and manage.

SMaWS Security layer is responsible to provide a secure communication between the SMaWS Manager and the Web Services that it intends to monitor and manage. Likewise, it is responsible to ensure integrity and confidentiality in both SMaWS manager and managed Web Services.

4.1.3 SMaWS Subsystem

The focus of this section is to describe the subsystem dependencies and interfaces within the overall SMaWS System (that represents entire scope of development effort.). A high-level overview of the entire framework is depicted in Figure 4:2. It shows subsystems with their interface dependences. The major subsystems are SMaWS Agent, SMaWS registry, and SMaWS Manager. The SMaWS Agent uses the SMaWS registry to store information about a given Web Service, which is later retrieved and used by the SMaWS Manager application. The SMaWS Manager application uses the retrieved information to establish a connection between the SMaWS Manager and the SMaWS Agent subsystem.

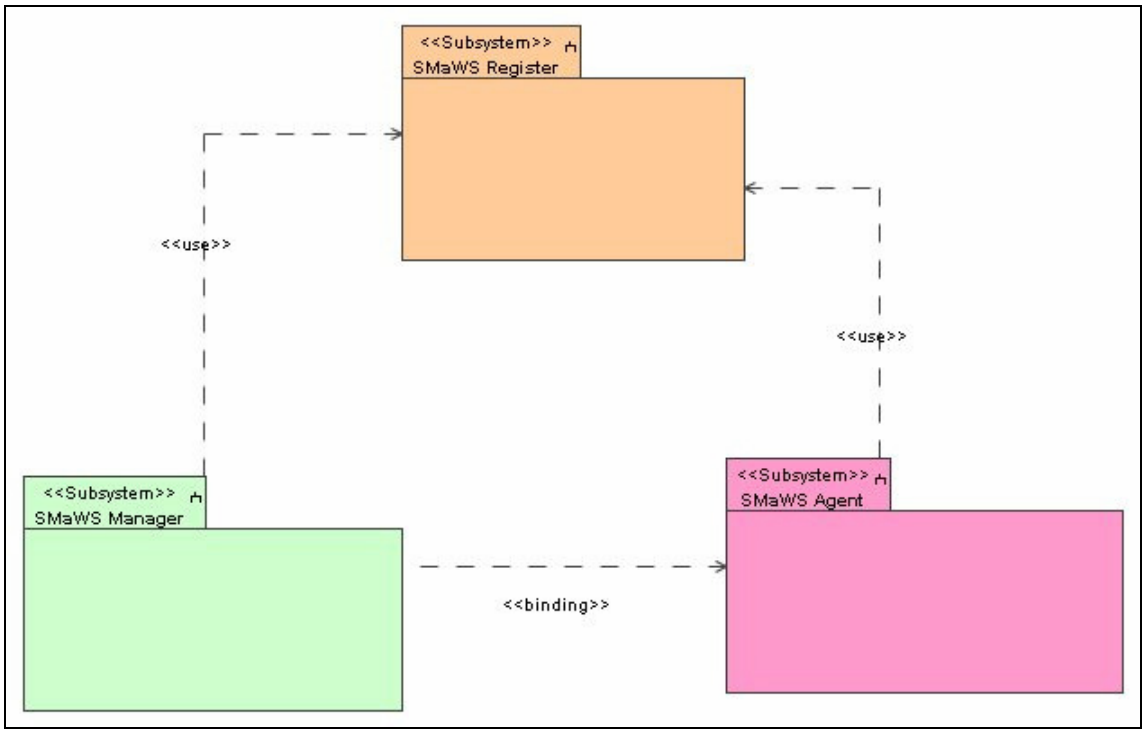

Figure 4:2 Main SMaWS Subsystem Dependence View

4.1.3.1 SMaWS Agent Subsystem

The SMaWS Agent Subsystem represents the core of the SMaWS framework, and it is embedded in the Web Service application. It has the responsible to provide management capability to the Web Service, and to expose aggregated measurement data to the SMaWS Manager.

SMaWS framework is compatible with old application server, as the Web Service Agent subsystem is inserted inside the Web Service application. If the SMaWS Agent was running external to the Web Service application, more effort shall be required to adapt or enable old application Server to have adequate management capability to manage the Web Service it host. Figure 4:3 shows the component view of the SMaWS Agent.

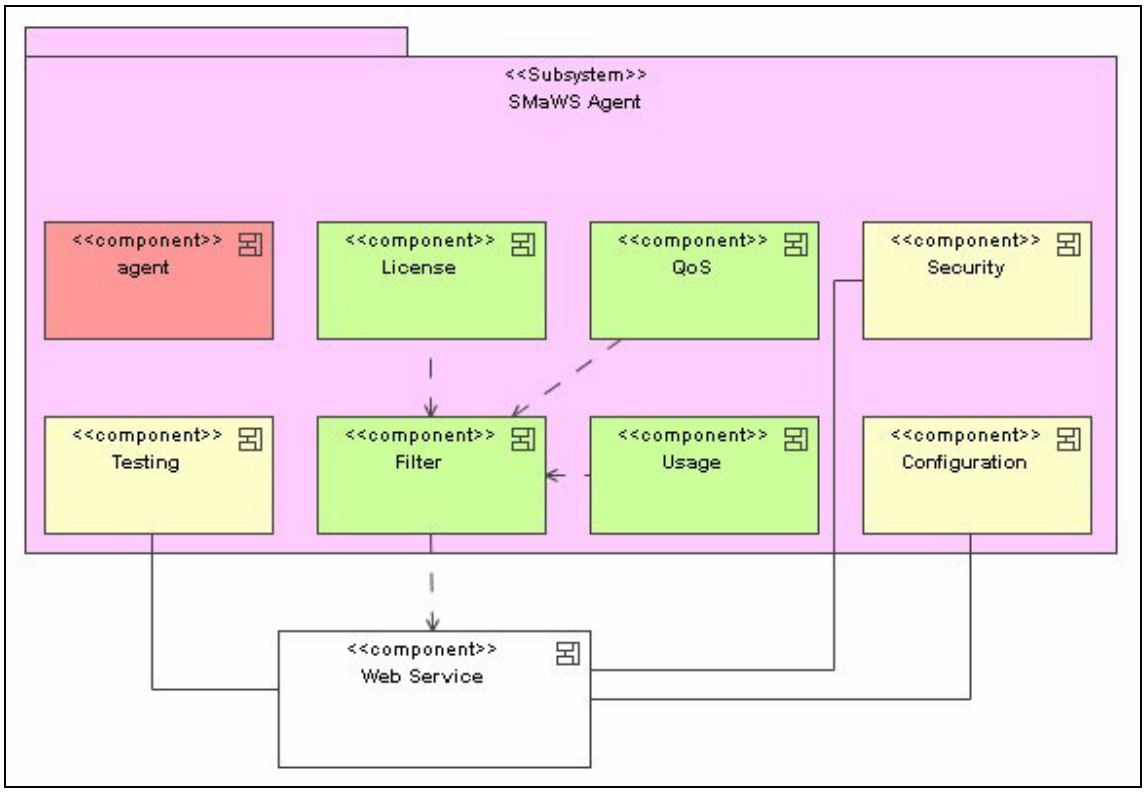

Figure 4:3 SMaWS Agent Component View

The SMaWS Agent subsystem consists of the following component:

- **Filter**: - The filter provides the data extraction functionality required to obtain data at runtime from the Web Service.
- **QoS Module**: **-** It is responsible to aggregate the measured QoS metric data provided by the filter module into a more meaningful context for the administrator. Examples of such aggregates are average response time since start-up and since the last 10 minutes, total number of requests, etc. It also provides information about Web Service Resource status,
- **Usage Module**: - It is responsible to monitor, and analyze the usage activities, such as information about the Web Service consumer, and the services they have requested.
- **License Module**: - It is responsible to analyze the license use by the Web Service consumer applications.

- **Testing module**: - It is an entity responsible to provide means for running test cases within the Web Services subsystem.
- **Configuration module**: - It is responsible to provide access to the Web Service configuration data. It is also responsible to provide Web Service Identification information such as Web Service name, Host name, Services, and Web Service description.
- **Security module: -** It is responsible to provide definition of security policies, users and access rights, activation and deactivation of security services within the Web Service subsystem.
- **Agent module:** - Agent module during the Web Service deployment registers the Web Service information and agent address in the SMaWS registry subsystem. The agent provides the means for the SMaWS Manager subsystems to extract result of the analysis from the respective modules.

4.1.3.2 SMaWS Registry Subsystem

The SMaWS registry is a repository where Web Service providers can published the information required by the management application to communicate with the Web Service agent. It provides a place were the management application such as the SMaWS Manager Subsystems can retrieve information about the Web Service it intends to monitor and manage.

SMaWS registry is design to address the bootstrapping problem that is how to inform the remote management application (client) about the details of the Web Service it intends to manage.

The SMaWS Registry is a database that maps names of Web Service to the encrypted Web Service agent address as well as the encrypted key use to enable a secure communication between the SMaWS Manager application and the managed Web Service

application. The entity *name* holds the public name of the Web Service application that is known to the public, in this case the SMaWS Manager application.

As compare to most registries, which are meant to address the bootstrapping problem such as the RMI registry in the Remote Method Invocation (RMI) framework [49, 50]. The RMI registry maps a name to a remote object value. While, the SMaWS Registry consists of three entities, which are name, remote object value and encryption key value. Secondly, the SMaWS registry contains information that could be accessed by the management application that does have the authorization to access them. In this case, the SMaWS Manager application is supposed to have a pair of asymmetric keys to verify and decrypt the content of the SMaWS registry map for a given Web Service public name.

4.1.3.3 SMaWS Manager Subsystem

It is responsible to provide the user-interfaces that enable administrator to discover, monitor, manage, and configure Web Services within an enterprise. It is also responsible to collect and present aggregated management data from a set of SMaWS agent in a comprehendible context to the administrator.

The SMaWS Manager subsystem is made up of the following set of composite component: Model, View, Controller, and adaptor as shown in Figure 4:4. The design of the SMaWS Manager is based on the Model View Controller architectural pattern [29, 36]. This approach divides a system that requires a human–computer interface. It organizes an underlying set of information into three components set, which are Model, View, and control.

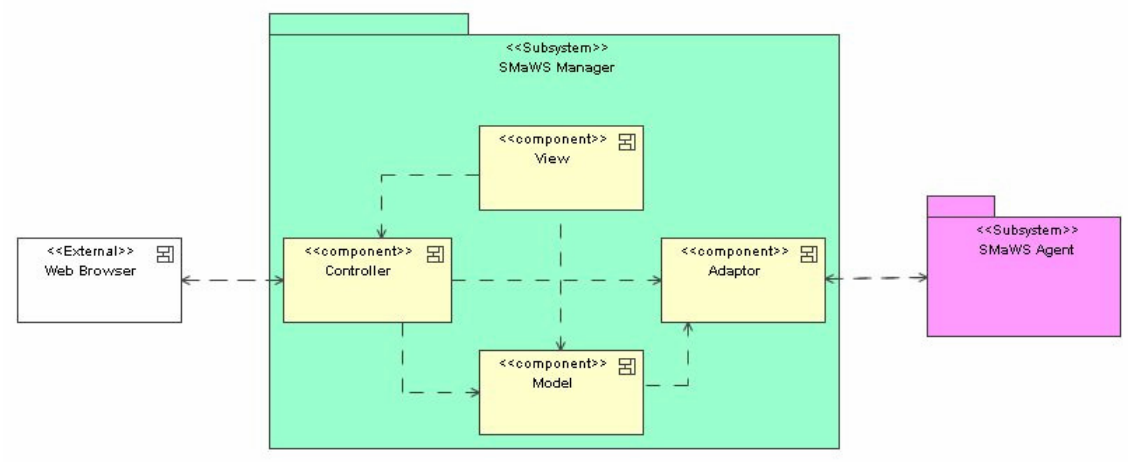

Figure 4:4 SMaWS Manager Component View

- **Controller:** The controller processes and sequences received http request made by the external Web browser. It determines the operation to perform and invokes the corresponding functional module within the model component to execute and response to the request.
- **Model:** The model encapsulates core data and functionality in the SMaWS Manager subsystem. It is independent of how the user interacts with the information. The SMaWS Manager Model component provides a set of operational modules that provide a one-to-one mapping with the web pages. The model operational modules call upon the Adapter component to establish connection to the SMaWS Agent, which uses a management protocol to expose and retrieve monitoring data it needs. The operational model after executing the user request delegated to them by the controller, it later calls on a specific view corresponding view modules to display the response. Operational Modules include:
  - **Login Form:** It executes the login operation requested from the login web page.
  - **Web Service Monitoring Form:** It is responsible for displaying the performance status of all Web Services assign to the register user. It is also responsible to display details about the Web Service identification information, and monitored QoS metric data.

- **Web Service QoS Management Form**: It is responsible to enable user to set QoS parameter.
- **Web Service Configuration Form:** It is responsible to enable the user to view and set changes to the Web Service configuration data.
- **Web Service License Management Form**: It is responsible to enable user to monitor the license used by the Web Service consumer application.
- **Web Service Usage Analysis Form:** It is responsible to enable users analysis the activities of the Web Service consumer application.
- **Security Configuration Form:** It is responsible to monitor and set Web Service security policy.
- **Web Service Testing Form:** it is responsible to enable user run test cases within the Web Services.

The SMaWS Manager model also provides a set of data model that includes:

- **User Data:** provides information about registered user. For example information about the user credentials, the list of Web Services assign to the user.
- **Web Service Data:** stores information about a given managed Web Services.
- **Web Service list Data:** stores a list or Web Services information instances, as well as provide functionality to retrieve specific Web Service.
- **Verification Manager:** provides authentication, encryption, and decryption services for the SMaWS Manager.
- **View**: The view defines the way the information are presented to human, and the acceptable set of manipulation capabilities. The view component receives calls from the operation modules of the component model to retrieve and display specific data to the user.

- **Adaptor:** The adaptor is responsible to establish communication with the SMaWS agent embedded in the Web Service application. It enables operation modules to set as well as retrieve monitoring and management data from the SMaWS agent in order to serve user requested information.

SMaWS Manager Application represents an instantiation of the SMaWS Manager layer. It is a Web-based application that runs on an Application Server (Web Server). It acts as a middle layer between a request coming from a Web browser or other HTTP client, and other Web Services application, which might be running on the same or remote application Servers.

As compare to other management system such as Application Server administration console, the SMaWS Manager Application designed in this work do not provide features to manage the Web Service host environment, as well as they lack features necessary to control the whole life of the Web Service, that is, to deploy, to stop, and un-deploy Web Services.

4.2 SMaWS Security Architecture

4.2.1 Design consideration

In the SMaWS infrastructure, Web Services are expected to register their SMaWS agent when deployed in a SMaWS registry. The SMaWS Manager application has to find and retrieve information about a given Web Service SMaWS agent, which it requires to establish connection with the Web Service. An attacker who controls the registry may compromise the confidentiality and integrity of the SMaWS Manager application, the Web Service, and the information transfer between the SMaWS manager application and the Web Service application.

In this section, our goal is to present a security architecture that addresses the following risk:

- **SMAWS Manager Application confidentiality**: - disclosure of information from the management application to unauthorized application. For example, information that is sent by the management application should be access only by the Web Service it intends to monitor and managed.
- **SMaWS Manager Application integrity**: - corruption, impairment, or modification of the management application by unauthorized application. For example, the information that is retrieved by the management application about the Web Service it intends to connect to should not lead to malfunction within the management application itself.
- **Web Service confidentiality**: - disclosure of information that is sent by the Web Service to unauthorized application. For example, information about the Web Service agent register in the SMAWS registry should be accessed only by the management application that has the authorization.
- **Web Service Integrity**: - corruption, impairment, or modification of the Web Service application to by unauthorized application. For example, the Web Service application should not allow unauthorized management application, which might belong to an attacker to perform or invoke operation that could, damaged the Web Services.

4.2.2 Security architecture

The security architecture in SMaWS is designed to ensure that a Web Service can securely send his address to the SMaWS Manager application. The SMaWS manager application can determine that the Web Service SMaWS agent information it has retrieved comes from the Web Service it intends to monitor and manage. The design also enable the Web Service to ensure that only authorize SMaWS Manager Application can monitor and managed the Web Service.

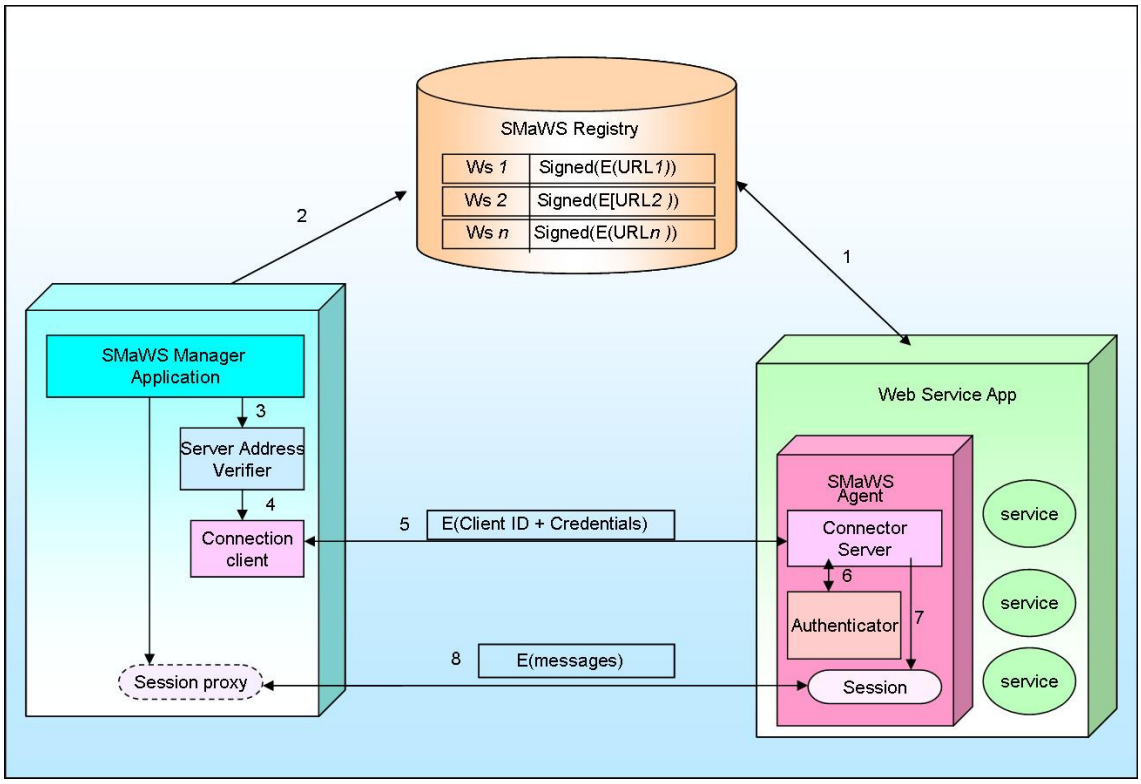

Figure 4:5 SMaWS Security Architecture

The sequences of steps are as follows:

1. Web Service Application generates a symmetric key it uses to encrypt the connector server URL address. This will ensure that less computerization process will be need (that is less overhead) to encrypt the large content of data that make-up the URL address The Web Service application uses the SMAWS manager application public key to encrypt the generated symmetry key. The Web Service uses its private key to sign the encrypted key. It later stores both the encrypted and signed information in the SMaWS registry.
2. The SMaWS Manager application use a Web Service's public name to requests and retrieves both encrypted SMaWS Agent connector server address of the Web Service and the signed key from the SMaWS registry.

3. Server Address Verifier uses the Web Service public key to verify if Web Service information, it has retrieved comes from the specified Web Service it intends to monitor and manage. That is, it will try to unsigned the signed and encrypted symmetric key.
4. If the signature is valid, the SMaWS Manager Application uses its private key to decrypt the encrypted symmetric Key. It later uses the decrypted symmetry Key to decrypt the encrypted Web Service Agent address, which is used to establish a communication channel with the Web Service Connector Server.
5. The SMaWS Manager application uses the decrypted symmetric key to encrypt the user credential. The SMaWS Manager application authenticates the user by forwarding his/her encrypted credential via connection client to the Web Service authenticator.
6. The Connector server hands over the credentials of the user to the authenticator.
7. If access is granted, the connector server creates a service session that is dedicated to the user.
8. The SMaWS Manager requests and SMaWS agent responses are made through the service session proxy.

To conclude, the Web Service send the management application its connector server address by using a secure channel based on both symmetric and asymmetric cryptography. The Web Service then uses the secure channel to exchange a symmetric secret key with the SMaWS Manager application. It later uses this symmetric key to provide a secure conversation using a symmetric cipher via the service session.

Chapter 5

SMaWS Implementation and Usage

5.1 JMX Overview

Many concepts in SMaWS are applicable across both .NET and J2EE platform, in spite of the fact that SMaWS implementation targets J2EE platform. This is because over the last years, Java has been consolidated as an interesting language in the network programming community. This is largely due to the design of the Java language that includes, among others, important aspects such as portability, architecture neutrality of Java code, and multithreading facilities.

The technology used in SMaWS framework for managing the instrumentation of the Web Service application is Java Management Extensions (JMX) [30, 45], which is an open technology that offers a lightweight, standardized way for managing Java objects. Java Management Extensions (JMX API) [30] is used as the core communication and management infrastructure. The strength of JMX includes its ability to provide a great deal of flexibility to management entities that can be constructed and modified dynamically, as well as enable management operations during runtime.

JMX API specified in JSR 3 [JSR 3] defined a way to create named objects called MBeans (managed beans) or managed objects, and to host those objects in a repository called the MbeanServer. Access to these objects is then exclusively done through the MbeanServer, which allows runtime deployment and manipulation.

MBeans have attributes (properties), operations (methods), and can emit notifications (events). The API provides services, which are themselves MBeans, and allows MBean attributes (properties) to be monitored periodically; notifications to be sent at a scheduled

time or times; new MBeans to be created from dynamically loaded classes; and relations to be defined between MBeans.

The JMX specification (JSR 3) does not provide a standardize means to access the Java based agent remotely. The Java Specification Request (JSR 160) [31] fills this gap by extending the JSR 3 to provide remote access to JMX enable application. JSR 160 provides a mechanism for remote access, which is based on the notion of connectors [44]. A connector provides remote client with API that makes remote access to JMX-based agent similar to local one. RMI connector is the only connector that is defined and presented in all implementations of JSR 160 specification. RMI connector is based on Java Remote Method Invocation (RMI) [50, 54] that defines RMI with two standard transports protocols, the Java Remote Method Protocol (JRMP), and the Internet Inter-ORB Protocol (IIOP) [49].

As compare to other technologies such as Simple Network Management Protocol (SNMP). JMX has the ability to provide a two-way communication that enables management applications to get data from managed resources, and send commands back to the managed resources when action is needed. SNMP provides only a one-way communication from the managed applications to the management application.

5.2 Monitoring Web Service Performance

Performance metrics are collected over time and allow us to view the overall health and performance of the Web Service. The SMaWS Manager is used as a tool to pull the continuously measure and aggregate QoS metric data from the SMaWS agent only when the SMaWS browser is open. The SMaWS Manager is configured such that when open, it enable a Minute-by-Minute view of the metrics data. This approach enables us to monitor Web Service much closely especially when a Web Service experiences heavy load.

5.2.1 Web Service QoS Metrics

The Performance section of a Web Service's view screen gives an overall view of how a given Web Service is performing. The following table defines each of the metrics that are collected for a Web Service.

Table 1 Web Service QoS Metric

Metric	**Value**
Total Requests Count	The total number of Web Service requests since startup.
Total Request Count since last 10 minutes	The total number of Web Service requests since last 10 minutes.
Request Count being Process	The total number of Web Service requests currently being process.
Average Response Time	The average Web Service response time in seconds for total Web Service requests since startup.
Average Response Time in last 10 minutes	The average Web Service response time in seconds for total Web Service requests since last 10 minutes. If no requests are sent during an interval, this field is left blank.
Throughput Value	Number of Web Service requests processed per second
Success Count	The total number of successful Web Service invocations.
Failure Count	The total number of failed Web Service invocations.
Maximum Response Time	The maximum Web Service response time in seconds for a Web Service request since startup.
Minimum Response Time	The minimum Web Service response time in seconds for a Web Service request since startup.

Authentication Success Count	The total number of successful authentication
Authentication Failure Count	The total number of times an authentication failure occur
Uptime	It represents the total amount time the Web Service has been running since deployed.

5.2.2 Performance Graph

The Web Service Performance graph provides a visual view of the performance metrics. The graphic represents the percentage of the Web Service performance value that is the percentage of average response time of the Web Service since last 10minute with respects to the SLO (Threshold). The graph is made up of a cylinder, which is filled with three different colours (green, yellow, and red) to represent available, failing, failed.

$$WsPerformanceValue = \frac{\text{ThresholdValue}}{\text{Last10MinAvgResponseTime}} * 100\%$$

If the performance is more than 70% to 100%, it is assumed that the service is available (up), as illustrated in figure 5:1. It means that the service is performing properly in processing the incoming requests, the time duration for processing the incoming requests is still within the range of the threshold (SLO) defined by the Web Service provider.

Figure 5:1 Service is available (Up)

If the Web Service performance value ranges between 40 to 70 percent, then service is still running but the average Web Service response time to the incoming requests is rather slower, or below the threshold (SLO) defined by the Web Service provider. The administrator need to check both the systems where the service runs, whether it is overloaded, and the service itself, whether there are some errors occurred. This condition is illustrated in figure 5:2.

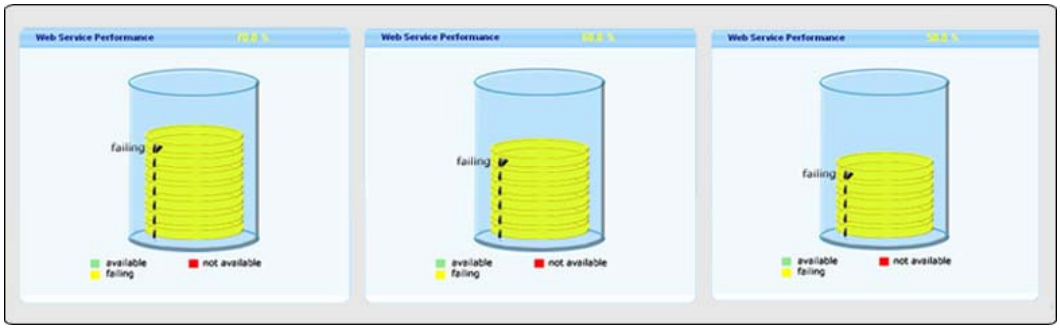

Figure 5:2 Web Service is failing

If the Web Service performance ranges between 0% and 40%, it is assumed that the service is not available (down), as illustrated in figure 5:3. The average Web Service response time for the incoming requests is too high, and far slower than the accepted SLO defined by the Web Service provider. This condition is unexpected to happen, because the administrator inquired to do responsive action, such as check the healthiness of the system, and check whether there are errors that occurred in the service during execution when the services is failing.

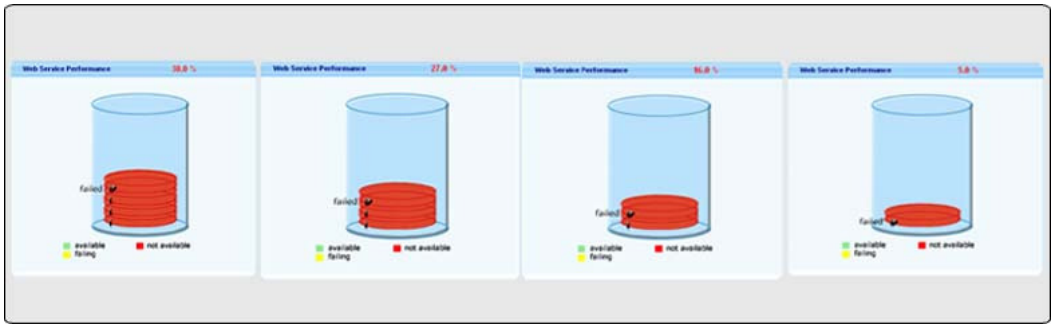

Figure 5:3 Web Service failed (Down)

5.3 SMaWS Manager Application

One of the main aims of developing SMaWS manager was to overcome the problems that are associated to desktop applications such as the need for local installation on each administrator's computer that wish to monitor their Web Services. A SMaWS Manager is a Web-based application that aims at providing a consistent interface that is easy to use, enabling the end-user to monitor, and manage their Web Service during run time. It can be simultaneously used by many user using any Internet connected computer with a standard Web browser from anywhere in the world, compare to traditional desktop applications, which can only be used by one user at a time while seated at a particular computer. It is flexible enough to be run from anywhere without the need for custom client software. In addition, since all computers nowadays come with built-in Web browsers, SMaWS Manager need no software to be installed, and can be transparently upgraded since all of the "software" is on the Web server.

SMaWS Management application enables the user to:

Retrieve and set QoS (quality of service) parameters, such as threshold.

- Monitor QoS metric data such as average response time since startup and since the last 10 minutes, total number of requests, etc.
- Monitor Web Service Resource status, such as Web Service name, Host name, Services, Web Service description, and status.
- Monitor and analyze the license use by the Web Service consumer applications.
- Configure the Web Service.
- Configure the Web Service security policy.
- Monitor and analyze the usage activities, such as information about the Web Service consumer, the service the requested for, if the request was successful or not.

The SMaWS Manager is based on the company Bayer's Java-based Web application framework called "NEAR Framework" which provide features to simplify development of user interfaces and session management. SMaWS Manager is a server side application that uses Java Server Pages JSP [37] for developing the views, Java Beans for Model and Java Servlet [38] for developing the controller modules. The SMaWS Manager's adaptor implementation uses JMX [30, 31] as the transport protocol, and it provides a control interface that allows transparent relaying of commands to the managed Web Service. The SMaWS manager provides scalable interfaces that allow users to monitor varied number of Web Services at an instance. During the development phase, the SMaWS manager is hosted on Apache Tomcat (formerly Jakarta Tomcat) [39], which is an open source web container.

5.4 Managed Web Services and Host System

Web Services used in this thesis are based on Apache Axis (Apache eXtensible Interaction System) [40]. Axis is a popular open-source XML based Web Service framework. It consists of a SOAP engine and standard functionality for deploying Web services that are implemented in Java. Once the Web Service application is deploy, the Axis runtime environment receives Web Service invocation and delegates the Web Service requests to the Web Service implementation class.

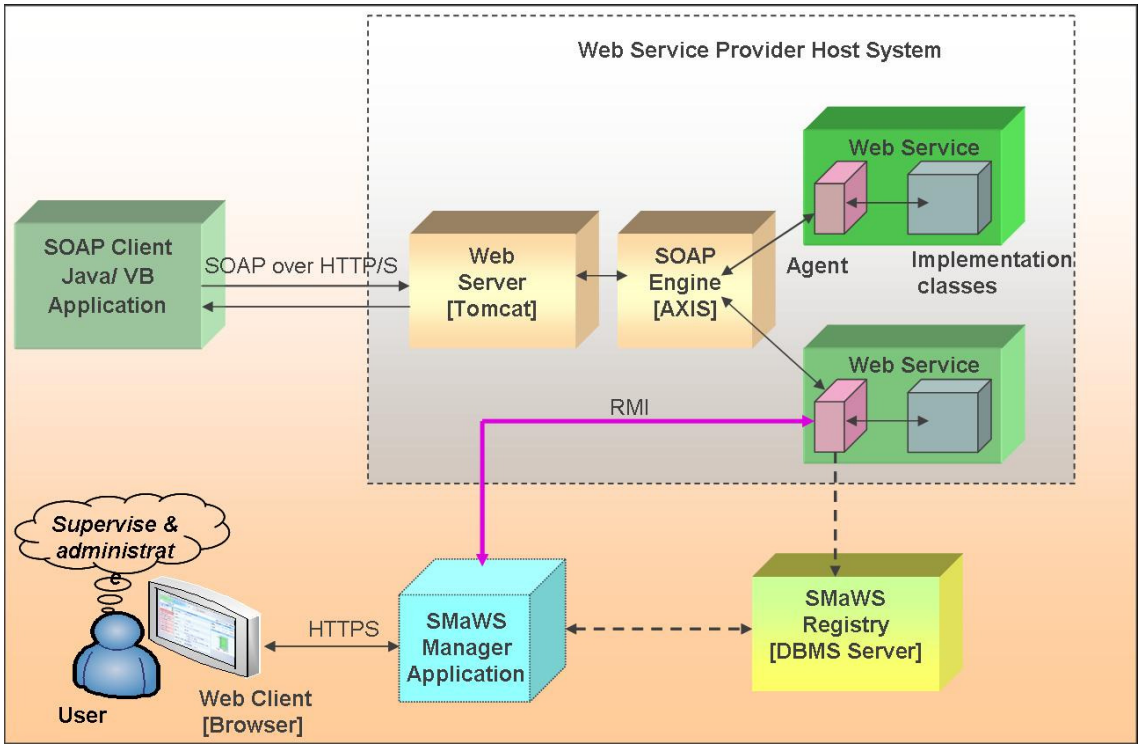

Figure 5:4 Managed Web Services and Host system

Figure 5:4 shows how a Web service consumer requests in the form of soap message are transported over a network via HTTP. A Web server such as Tomcat, which delegates the processing to Axis SOAP engine, receives the request on the Web service provider side. The SOAP engine invokes the appropriate Web Service implementation class to process the message, and the Web server later sends the reply from the Web Service application to the Web service consumer application. The SMaWS Agent performs, which performs the monitoring activities, uses its filter module to intercepts both the request message and response messages that the given Web service has processed. The filter module extracts measure metrics data that are aggregated (e.g. response time to average response time) by other Web Service agent's modules.

The aggregated metric data are later sent to the SMaWS Manager Application when requested. Once the Web service operator logs-in and intends to monitor and manage

Web services, the SMaWS Manager application forwards the request to the SMaWS Registry, to retrieve the specified Web services connection information. The SMaWS Registry is implemented as a relational database. The SMaWS manager application later uses the connection information to establish a connection to the SMaWS Agent. This connection is based Java Remote Method Invocation (RMI) standard protocol. The SMaWS Application uses the connection to retrieve aggregate measured data from the managed Web service. It also uses the connection to send commands back to the managed Web Service.

5.5 SMaWS Agent Implementation

In order for a Web Service to be "manageable", the Web Service management domain must be instrumented with data collection and event detecting probes. The instrumentation consists of implementation codes that have to be included intrusively or non-intrusively with the managed Web Service. Non- intrusive instrumentation effectively means no change to the application code is required. While intrusive implementation means modifying the application in any way for instrumentation.

SMaWS framework enables intrusive instrumentation of the Web Service components, and therefore fine-grained view of the activities of components constituting the services. It provides instrumentation application programming interface (API) for Web service that are inserted into the Web Service during Web Service development. The empirical result indicates that additional instrumentation code causes an overhead less than 25%, which is acceptable. The SMaWS Agent instrumentation approach focus on solving shortcoming of existing Web Service instrumentation approaches by providing transparency for the Web Service developer.

Alternative approaches focus on realising non-intrusive instrumentation of the Web Service application in a transparent manner, for example by instrumenting the Web Service runtime environment. This approach enable Web Service component to be instrumented automatically during deployment. This instrumentation approach provides limited level of information details.

The SMaWS Agent Subsystem represents the core of the SMaWS framework and it is embedded in the Web Service application. It has the responsible to expose aggregated measurement data to the SMaWS Manager.

The SMaWS Web Service Agent is based on Java Management Extensions agent layer of JMX framework. The JMX agent layer is made up of an MBeanServer, which host Mbeans that represents manageable resources. MBeanServer allows runtime deployment of Mbeans, and all management operations performed on the Mbeans are done through interfaces on the MbeanServer. Each of the implementation of the Web Service agent modules are deployed as additional Mbeans in the MBeanServer. Figure 5:5 show an overview of SMaWS Agent Class Diagram

Figure 5:5 shows the relationship among the essential set of classes and interfaces found within the Web Service Agent subsystem. Different colors have been used to distinguish the various classes found in each of the main modules that make up the Web Service agent subsystem as depicted in figure 5.5. The *"WebServiceAgent.class"* is based on the MbeanServer, and it represents the agent main module, which is used for the integration of various sub modules that make up the web Service agent subsystem. Figure 5:5 shows each of the various sub modules that are integrate directly to the *WebServiceAgent.class* such as the *QoSMonitoring.class,* and they are associated to an interfaces (MBean) that describe their manageable operations. Details of some of these modules classes shall be described in the next sub sections.

Tables 2 give a brief explanation of all essential classes and interfaces found the Web Service Agent implementation.

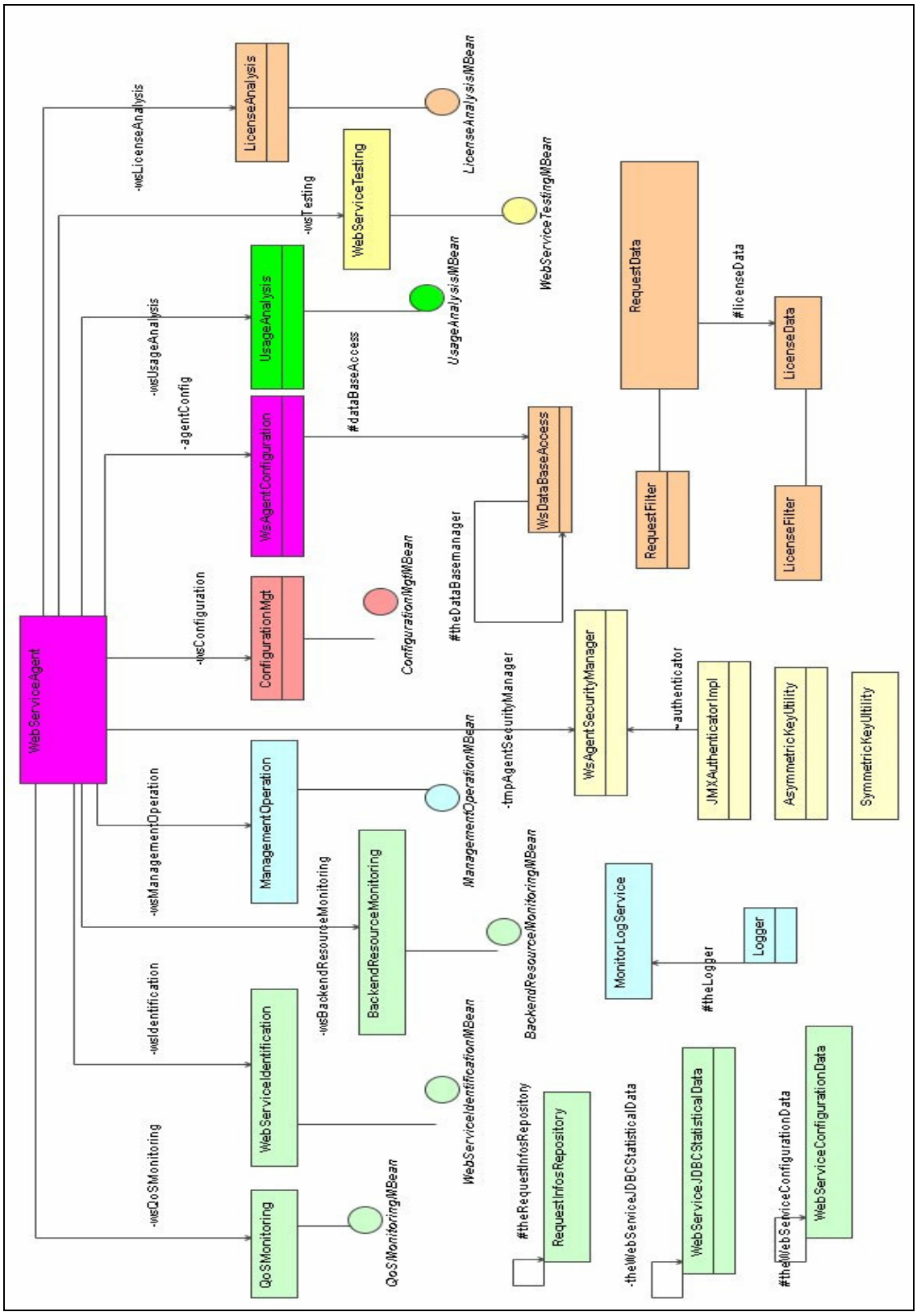

Figure 5:5 Overview of SMaWS Agent Class Diagram

Table 2 Explanation of the SMaWS Web Agent Implementation classes and interfaces

Classes/Interfaces	**Implementation of**	**Explanation**
WebServiceAgent	Agent Module	Provides operation to enable integration of sub modules and access to registered managed resources
WsAgentConfiguration	Agent Module	Provide operations for the configuration of the Web Service Agent sub system.
QoSMonitoring	QoS Module	Provides operations for the extraction of monitoring data
QoSMontoringMBean	QoS Module	Indicate manageable operation for monitoring
WebServiceIdentification	QoS Module	Provides operations for the identification of Web Services
ManagementOperation	QoS Module	Provides operations for the control of monitoring activities
ManagementOperationMBean	QoS Module	Indicate manageable operation for monitoring
BackendResourceMonitoring	QoS Modue	Provides operations for the extraction of backend monitor data
BackendResourceMonitoring-Mbean	QoS Module	Indicate manageable operation for monitoring
Logger	QoS Module	Provides operations to log activities of the Web Services
MonitorLogService	QoS Module	Ensures that only one instance of the Logger class is use as a global point of access.

RequestFilter	Filter Module	Provide operations for the extraction monitored data.
RequestInfoRepository	Filter Module	Provide operations to store and retrieve stored requests information
LicenseFilter	Filter Module	Provide operation for extracting License information from Web service consumer requests.
WebServiceTesting	Testing Module	Provide operations for testing Web Service
WebServiceTestingMbean	Testing Module	Indicate manageable operation for Web service testing module
LicenseAnalysis	License Analysis Module	Provide operations for analysis the licenses use by the Web Service consumer application
LicenseAnalysisMBean	License Analysis Module	Indicate manageable operation for License Analysis module
ConfigurationMgt	Configuration Module	Provide operations to access the Web service configuration data.
ConfigurationMgtMBean	Configuration Module	Indicate manageable operations for the configuration module
WsAgentSecurityManager	Security Module	Provides operations for security management related to SMaWS
AsymmetricKeyUtility	Security Module	Provides operations for asymmetric encryption and decryption
SymmetrcKeyUtility	Security Module	Provides operations for symmetric encryption and decryption
UsageAnalysis	Usage Module	Provide operations for accessing Web Service consumer usage activities.
UsageAnalysisMBean	Usage Module	Indicate manageable operations for Usage analysis module

Figure 5:6 illustrate manageable operations available in the SMaWS Web Service Agent subsystem for monitoring and management of Web Services. These operations are invoked by the SMaWS Manager application via the agent module within the SMaWS Agent subsystem. Manageable operations found within classes filled with the yellow – green color are meant for the monitoring of the Web Services, and while operations found in classes filled with the "gold "color are meant for the management of the Web services .

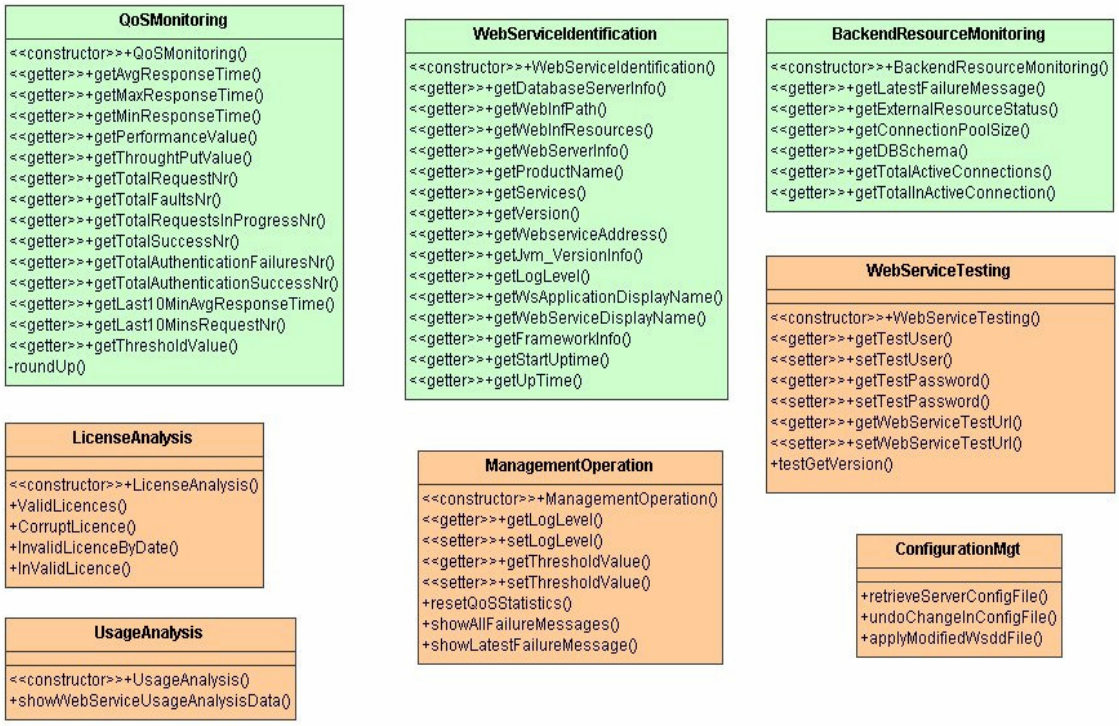

Figure 5:6 Examples of manageable operations for the monitoring and management of Web Services.

5.5.1 Agent module

The agent modules made up of the *WebServiceAgent. Class* and *WsAgentconfiguratio. Class.* The *WebServic Agent.class* act as a unified interface to a group of interfaces in the subsystems, and enables remote objects to access the subsystem using its interfaces to

communicate with the subsystem. It also shields SMaWS Manager Application (client application) from SMaWS Agent subsystems components. Figure 5:7 illustrates the initialization process of the Agent module.

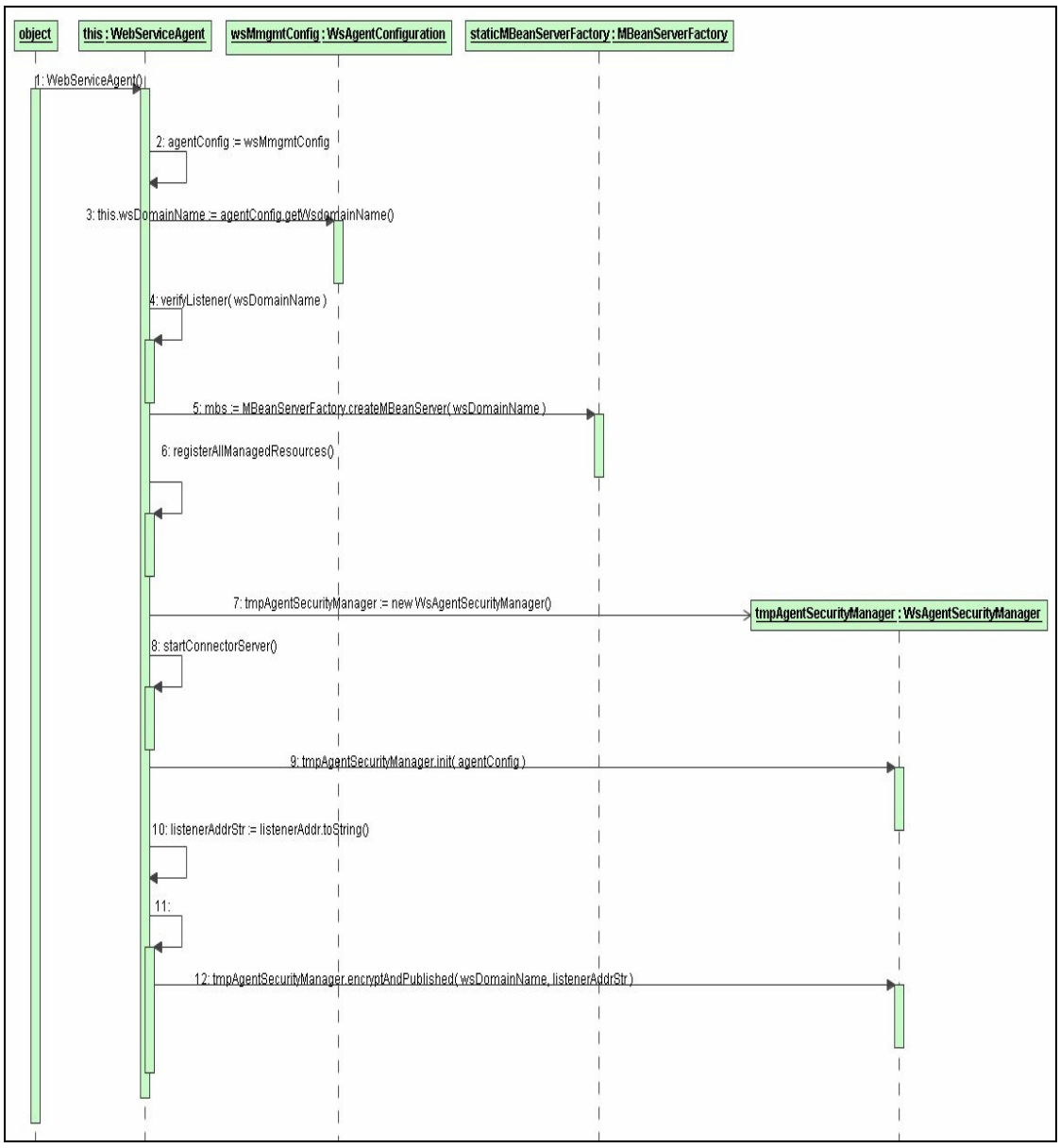

Figure 5:7 UML Sequential Diagram of the initialization process of the Agent module

In this figure 23, during the initialization phase of the Web Service Agent Subsystem following process takes place:

1. The instantiation and initialization process of the Web Service Agent Subsystems begins when the constructor of the Web Service Agent class is invoked
2. A reference of an instance of the Web Service Agent configuration class is assigned the Web Service Agent.
3. The public name of the Web Service is assigned to the given instance of the Web Service agent (listener).
4. Verification of the domain name is done to ensure no listener with the given domain name is already loaded in the run time Java Virtual Machine (JVM).
5. An MbeanServer with the specific Web Service Agent domain is created
6. Web Service agent service modules such as the QoS module, and licenses analysis module are registered to enable access to their manageable operations.
7. Security manager module of the Web Service agent is instantiated.
8. A connection server component is created and activated to start listening to RMI based remote clients connections.
9. The Web Service Agent's security manager module is initialized.
10. URL address of the Web Service agent is retrieved from the connector server.
11. The Web Service Agent invokes the security manager module, to encrypt and publish the Web Service address in the SMaWS Registry.

5.5.2 SMaWS Agent Address.

The Web Service Agent Address is the address used by the SMaWS Manager Application to locate the Web Service they intend to manage. This address actually represents the address of the Connector server module that is responsible to establish an RMI connection with remote client connection module created in the SMaWS Manager Application. In figure 5:4 above, the connection between the SMaWS Manager Application and SMaWS Managed Web Service application is based on RMI protocol.

Compare to other remote communication protocol such as CORBA [52], RMI protocol enable faster transfer of data, and induced little overhead. For real-time monitoring

infrastructure, it is ideal to have a situation where the time between when an event occurs in the managed Web Service and the time information about the event is display on the admin console is zero. Unfortunately, this is not possible due to the time required by each protocol for the transfer, serialization, and de-serialization of the event message.

Java RMI use optimized connection oriented communication protocols that is language specific. It is based on the concepts of subs and skeletons, which enables the RMI clients to invoke a method locally, while it is actually executed in a remote object. The following facts are involved in the operation of RMI infrastructure:

1. The port that the remote server is listening to is usually chosen arbitrarily at runtime and is decided by the JVM or underlying operating system.
2. The remote server usually does not listen to default 1099 or given port number specify by the developer.
3. The default port (1099) or specified port is usually used by RMIregistry, which is also a remote object.
4. The port that the remote server listens to is usually chosen arbitrarily at runtime and decided by the JVM or underlying operating system.
5. The client does not know which port the server is listening to, but it has a reference to an object of the stub that does know.
6. A stub is created and published in the RMIregistry by Remote server, and the remote client later retrieved this stub by using the Remote server public name.
7. Once the client gets the stub object from the RMIregistry, the Registry is not used anymore
8. The RMIregistry is used only to enable the remote client locate the Remote server.

The challenges I face while using the RMI protocol are as follows:

1. How to ensure that the address that I intends to send to a remote client application (SMaWS Manager) is the address of the Connector server (Remote Server) and not the address of the RMI Registry.

2. How to transport securely the stub object created at the connector server to the remote client.
3. How to avoid unauthorized remote client application use the public name of the connector server (that is name of the Web Service) to retrieve Connector Server's stub object from the RMIregistry.
4. How to ensure only remote client applications with authorization could access the address of the connector server.

The measures taken to address these challenges are as follow:

1. Avoid the use of RMIregistry, since RMI registries are insecure as they do not support SSL/TLS [55], and cannot carry out a remote client's authentication or prevent untrusted clients from accessing the Connector server stub.
2. Invoke the connector server address from the Connector server after it has exported itself to an arbitrary port on the host server machine.
3. Encrypt the invoked Connector server address; such that it can only be decrypted by authorize remote clients (see section 5.6).
4. Store the encrypted Connecter Server address in a SMaWS Registry, which is known to remote clients.
5. SMaWS registry will act as a repository that contains entries that maps the public name of the Web Service to a specific encrypted connector server address.
6. The SMaWS Manager application (remote client) uses the public name of the Web Service to retrieve the remote Connecter server address (also known as Web Service Agent address.)

5.6 SMaWS Security Implementation

The approach use for Web Service Management in this research enables the SMaWS Manager application to be loosely coupled to the each of the SMaWS Agent embedded in the Web Service it intends to monitor and managed. All information about the Web Service that are deployed within the enterprise environment are store in a common registry (SMaWS registry), and later retrieved by the SMaWS Manager application. This approach leads to more flexibility, and automatic detection of managed Web Services that are deployed.

In section 4.2.2, SMaWS security architecture of SMaWS was presented, and it is designed with the aim to allow a Web Service Agent that is embedded within a Web Service, to send his address to the SMaWS Manager application, as well as enable a secure communication between them. The implementation of this SMaWS security infrastructure was based on several aspects of the Java 2 Security that includes Java Cryptography (JCE) [46], and Java Authentication and Authorization System (JAAS) [47].

The Java Cryptography Extension (JCE) is a set of packages that provides a framework and an implementation of cryptographic algorithms for both Symmetric and Asymmetric cryptography. The JCE is subjected to US export restrictions, and it is a pluggable technology, which allows different implementation from many providers. In this thesis, the JCE that was used comes from the open source provider Bouncy Castle [56].

The Java Authentication and Authorization Systems (JAAS) [47] is a set of packages that provides a framework for authentication and authorization into codes. It provides the following functionalities: authentication (for user login) and authorization (for permission checks). It is a standard extension to the Java 2 Software Development Kit, version 1.3 (J2SDK 1.3), and is part of a higher version Java 2 Software Development Kit.

In Figure 5:8, shows the class diagram of the most important components SMaWS security module, which is made up of the following classes*: WsAgentSecurityManager.class, VerificationManager.class, AsymmetricKeyUtility SymmetricKeyUtility.class,* and *JMXAuthenticationImpl.class.* Table 3, gives a brief explanation of the various components found in the SMaWS implementation module.

The key pairs used by both the SMaWS manager and the managed Web Service are based on the RSA Algorithm. The *AsymmetricKeyUltility.class* interface *generateKeyPair()* is invoked by Web Service application and SMaWS Manager to generate a key-pair (public and private key) that is used for its digital signature and encryption. All of Key-pair generation starts with a *KeyPairGenerator*. The format of public and private keys are not `RAW`. The public key is in `X.509` format and the private key is in `PKCS#8` formats.

The *SymmetricUltiliy.class* interface *generateSymmetricKey()* is invoked by the Web Service agent to create secret key used for symmetric encryption. The generated secret keys are based on the Triple Data Encryption Standard (TripleDES) algorithms.

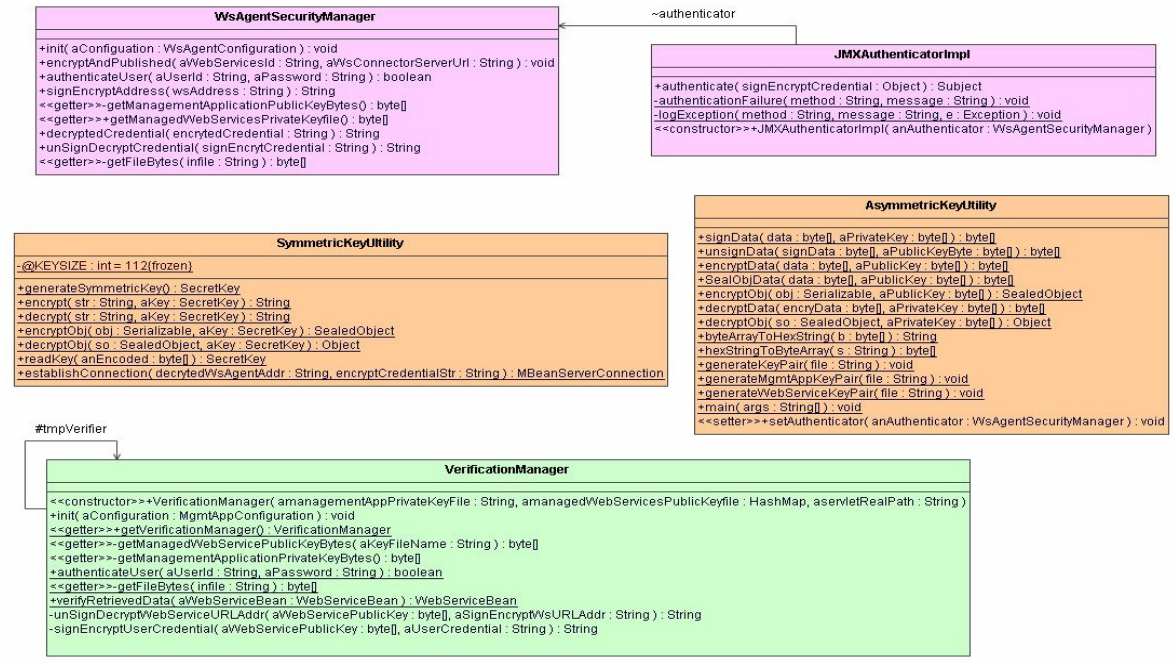

Figure 5:8 Overview of SMaWS Security modules

Table 3 Description of implemented components in the SMaWS security module

Component	**Implemented in**	**Explanation**
WsAgentSecurityManager-.class	SMaWS Web Service Agent Subsystem	Provides operations for the signing, encryption and publishing of the Web Service Agent address.
VerificationManager.class	SMaWS Manager Subsystem	Provides operations that retrieve and verify the download Web Services address from the SMaWS Registry.
JMXAuthenticationImpl.class	SMaWS Web Service Agent Subsystem	Provides operations that authenticate user's credential data.
AsymmetricKeyUtility.class	Both in SMaWS Web Service Agent Subsystem and SMaWS Manager Subsystem	This class provides operations for the creation of public and private key. It also provides operations for asymmetric encryption and decryption of data.
SymmetricKeyUtility.class	Both in SMaWS Web Service Agent Subsystem and SMaWS Manager Subsystem	Provides operations for the generation of secret key, which is use for the symmetric encryption.

The *WsAgentSecurityManager.class* components represent the *authenticator* module in the SMaWS agent security module. The *encryptAndPublish()*method provided by *WsAgentSecurityManager.class* component is invoked during the deployment of the Web

Service to encrypted, and published the Web service Agent connector server address in the SMaWS registry. The *WsAgentsecurityManager.class* component is referenced by the *JMXAuthenticatorImpl* module that delegates all authentications of the SMaWS Manager User credentials, before establishing a connection between both applications.

The *VerificationManager.class* module is part of the SMaWS Manager application. It represents the *Server Address verifier* component in the SMaWS Security architecture (see section 4.2.2), which is used to verify the signature of the Web Service agent information retrieved from the SMaWS registry.

Before a connection is established the SMaWS manager application invokes the method e*ncrpytUserCredential ()* in the *VerificationManager* module in order to encrypted the User information that has requested for connection to the connection between the SMaWS Manager application and the SMaWS based Web Service it intends to monitor and managed. At the Web Service, the SMaWS Agent ConnectionServer requests the *JMXAuthenticatorImpl* module to authenticate the user. The JMXAuthenticatorImpl module delegate the request to the *WsAgentSecurityManager.class* component that invokes both the d*ecryptUserCredential()* methods to decrypt the User credential, which is later forwarded to the *autheniticateUser ()* method. If the authentication process is valid, a *javax.security.auth.Subject* object representing the user is return, and associated with a dedicated session proxy that is created, and a proxy for it is return to the SMaWS Manager application. The dedicated session maintains the information about the specific user that has been authenticated. This dedicated session proxy is similar to a "ticket" in single sign-on architectures such as Kerberos [48]. Possession of a dedicated session proxy provides the user access to retrieve monitoring information and manage the Web Service as long as the session is considered valid by the Web Service SMaWS agent.

5.7 Usage Example

This section demonstrates the feasibility and usefulness of using SMaWS for the monitoring and management of Web Services. It describes the usage of SMaWS Manager Application within the enterprise Bayer Business Service GmbH to monitor two sample Web Services (that is, *Logbook Web Service* and *Customer Management Web Service*) and a Web Service consumer application (*ICS Patent plus*) on the development servers infrastructure. A brief overview of this enterprise applications is describes as follows:

- *Logbook Web Service:* This Web Service provides two services, which are log usage services, and log error-event service. Logbook Web Service clients applications are Web application that provide online laboratory services to research in the chemical industry Bayer AG.
- *Customer Management Web Service:* This Web Service offers the following services: authentication, authorization, access to customer data, access to the human resource data, and update of customer properties. These services are meant to be accessible to other applications that might require one or all of the services in other to go on with their processes.
    - The *Authentication service* involves validating username and password that have being sent by an application, which the user is trying to access.
    - The *Authorization service* checks for privileges of the user with a given identity. In the Customer management Web Services, this privilege implies the roles assigned to each customer.
    - *Access to customer information service* returns all customer information, lists of customer role for a given application, lists of customer properties for a given application, and lists of customer cost centres for a given application.
    - *Access to a user human resource data service* provides access to the customer information on the SAP system.

- *Update of customer property service* allows modification of customer's properties assigned to a given application.

- *ICS PatentPlus:* It is a Bayer intranet application to search in house patent databases. It offers users means to access information to million of Derwent patent abstracts, their legal status, and patent family data. This Web Service depends on the Customer Manager Web Service application to authenticate its users as well as determine their access right.

5.7.1 Web Service Monitoring

The first steps illustrated in figure 5:9, shows the screenshots of the login page that request Web service's operator to enter their credential and login. This ensures untrusted users do not have access to the systems.

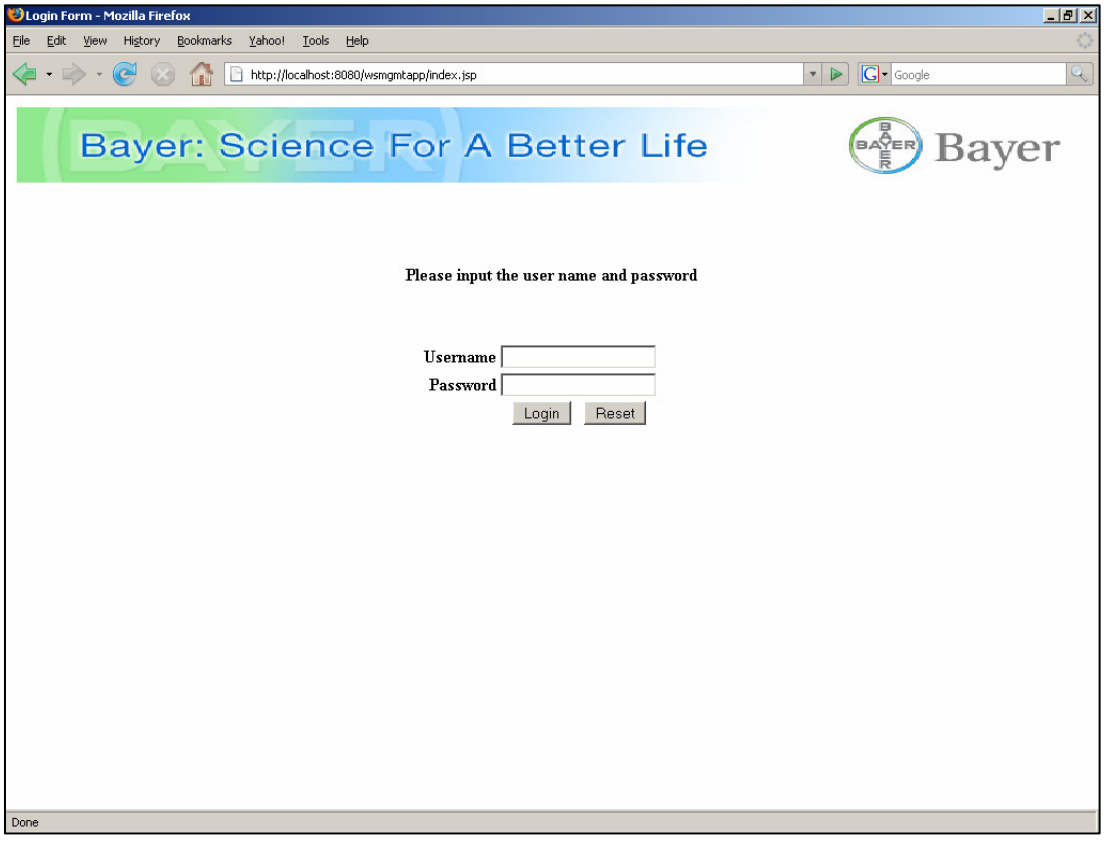

Figure 5:9 Login Page

Secondly, if the login is successful, the SMaWS Manager application determines the list of Web Services the operator is allowed to monitor and manage. It switches to display Web Service monitoring page if at least one of these Web Services are deployed.

In case, no Web service is deployed the SMaWS Manager application sends a feedback to the operator that displays a list of Web Services he is allowed to monitor that are currently not deployed.

Thirdly, if the operator is allow to monitor both of the *Logbook Web Service* and the *Customer Management Web Services Application*, which are deployed on the development server. The SMaWS Manger switches to Web Service Monitoring Page as shown in figure 5:10 screenshot. The User can click on the left navigation to see the details of each given Web Service deployed. The following information is displayed:

- List of deploy and undeploy of Web Services,
- Performance information of Web services,
- Identification information of Web Services,
- QoS metrics information of Web Services in the last 10 minutes and since start-up, see figure 5:11,
- Performance information of Web Service's external backend systems such as the Database system on which it depends, see figure 5:12.

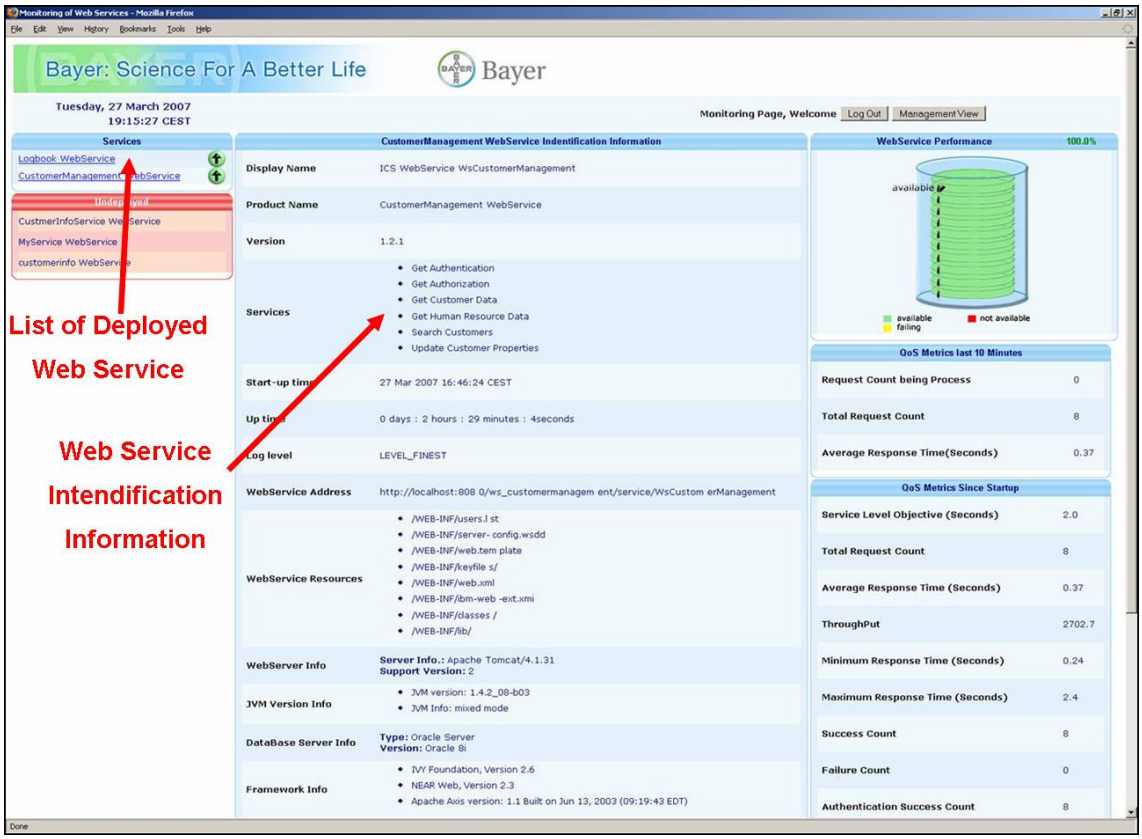

Figure 5:10 Web Service Monitoring Page

Figure 5:11 shows the QoS Metric data collected from the Customer management Web Service. Figure 5:12 shows the monitored information of the backend systems on which the web service is depends on. It shows that all the Customer Managements Web Service external resources are up and running.

Figure 5:11 QoS Metrics of a given Web Service displayed by the SMaWS Manager application

Figure 5:11 Web Service External Resource status

Fourthly, the operator could click the Management View button on the monitoring page and navigate to the management pages, depending on his access rights to toward the given Web Services.

5.7.2 Web Service Management

In the process of managing a given Web Service, the SMaWS Manager applications allows Web Service operator with the privilege to manage a given Web Service, the means to configure, analysis, and administrate the monitored Web Service.

In the management pages, the QoS Management view is display as default view, see figure 5:12. In this view, the user could set the QoS parameter of a given Web Service. In addition, the user could navigate to the following views:

- Usage analysis view, which displays information about activities of the Web Service consumer application, see figure 5:13,

- License analysis view, which displays license information used by the Web Service consumer application, see figure 5:14,
- Web Service Configuration view, which allows the user to retrieve the Web Service's *server-config.xml*, see figure 5:15,
- Web Service Testing view that display operation that the user can click on to test the web service. 5:16
- Security Configuration view (User Management view) that displays information about the list of Web service operator for the given Web Service, see figure 5:17

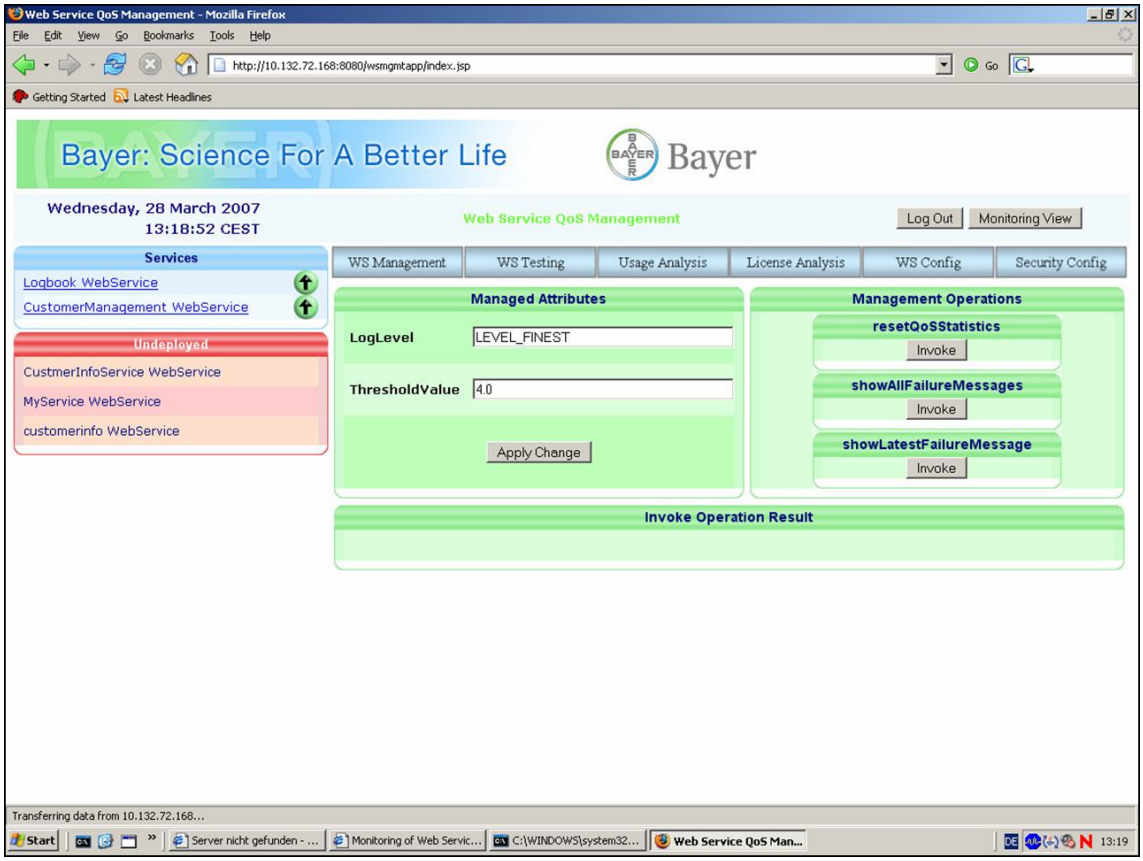

Figure 5:12 Quality of Service (QoS) management view for the Customer Management Web Service

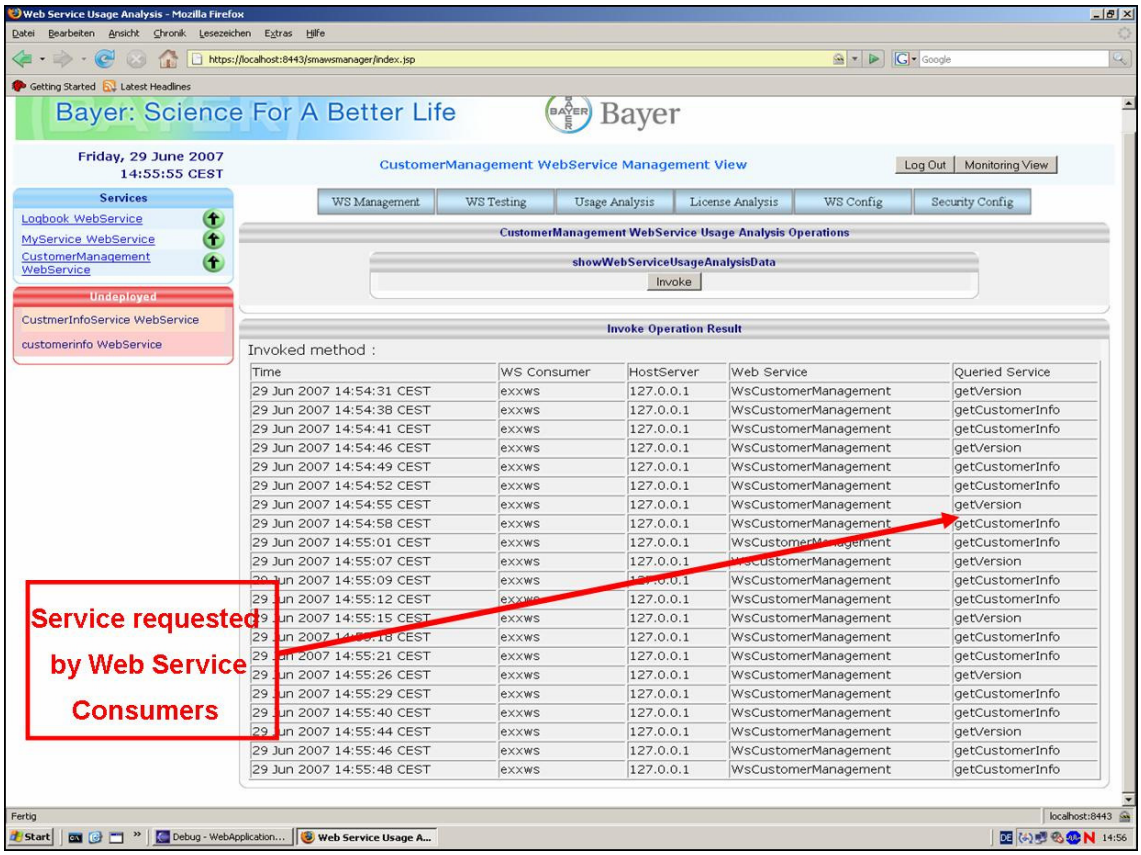

Figure 5:13 Usage Analysis view for Customer Management Web Service

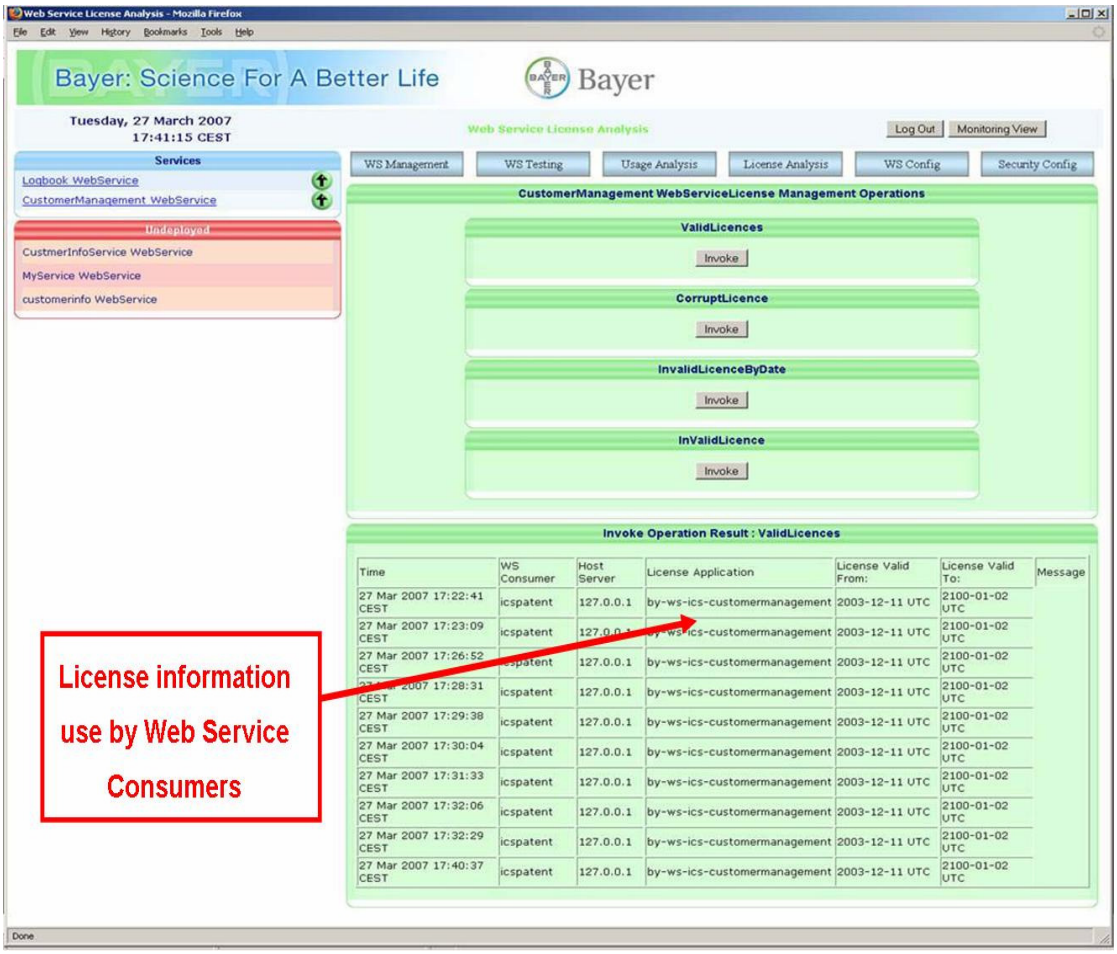

Figure 5:14 License Analysis view for a the Customer management Web Service

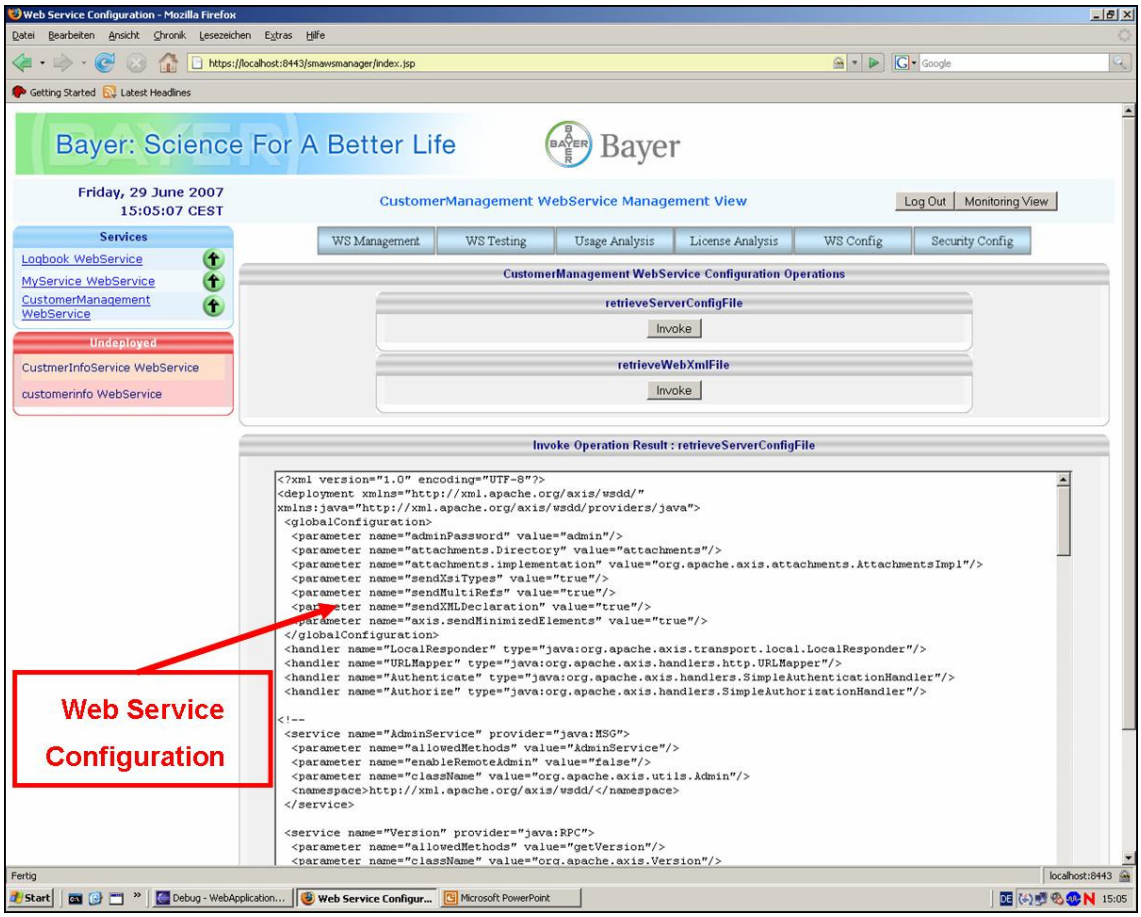

Figure 5:15 Web Service Configuration view for the Customer management Web Service.

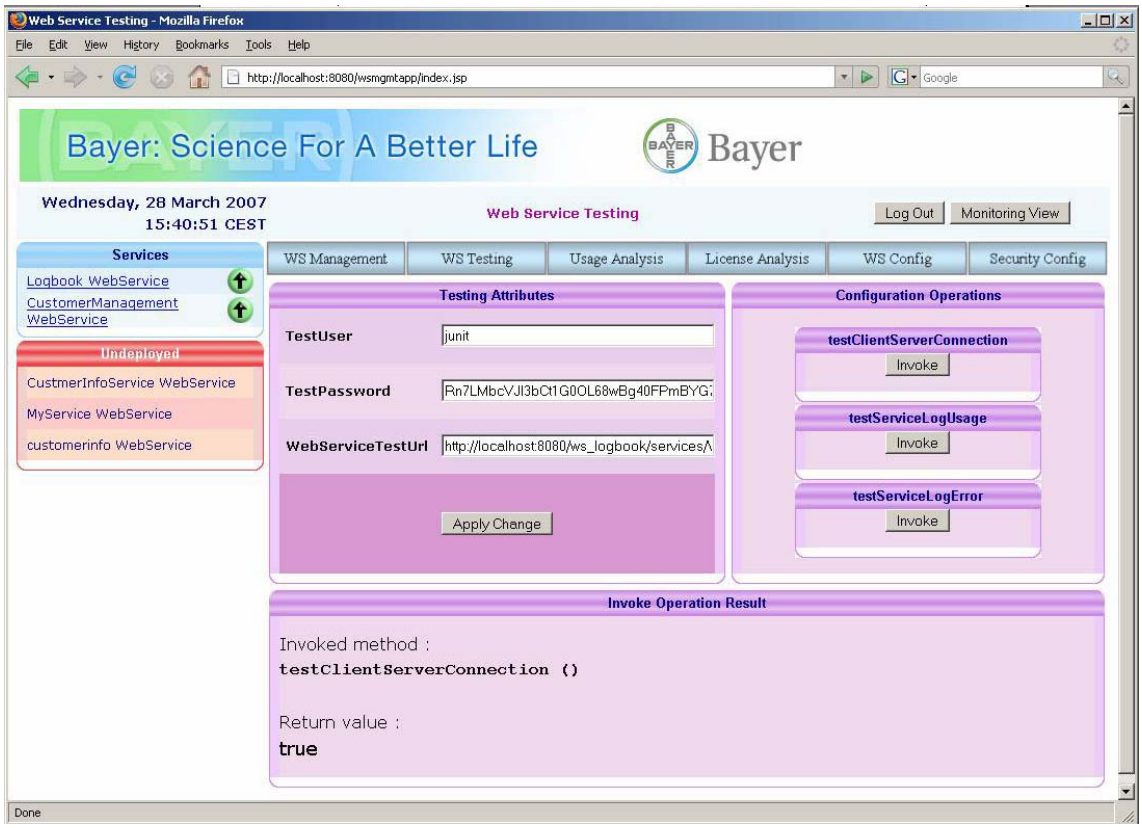

Figure 5:16 Web Service Testing view for the LogBook Web Service

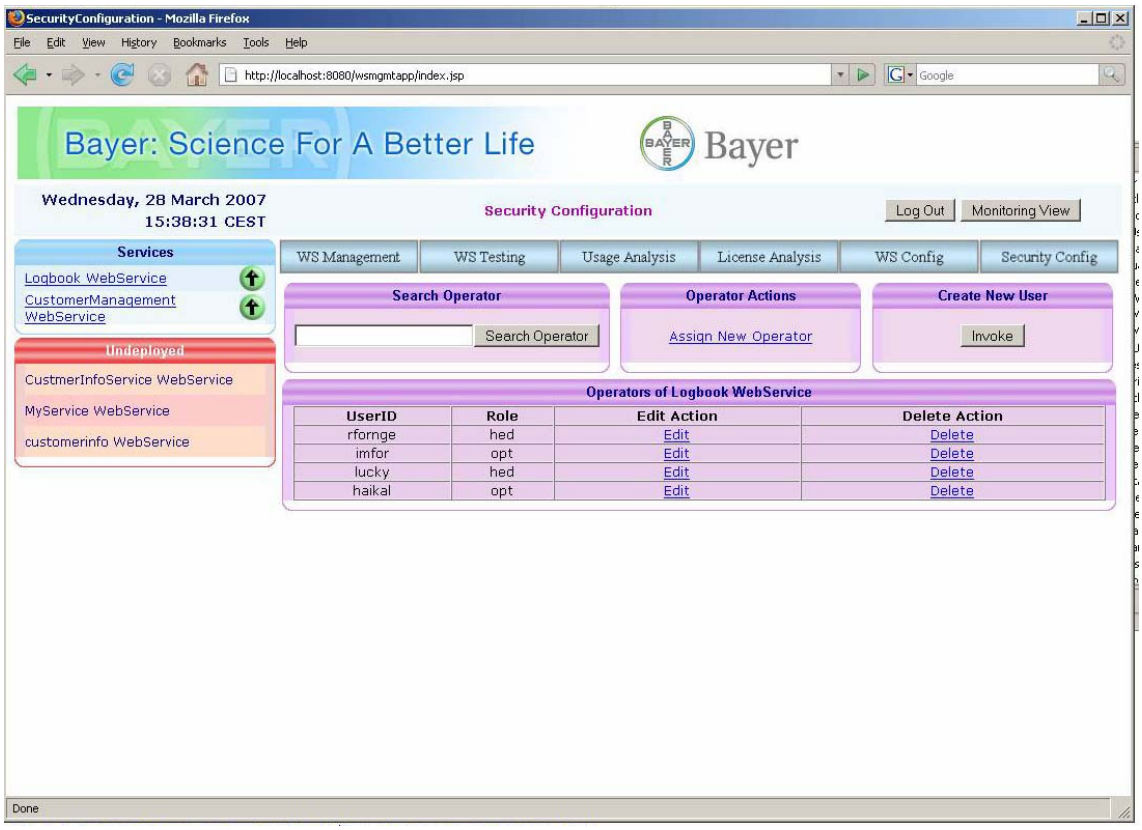

Figure 5:17 User Management view for Logbook Web Service

Chapter 6

Experimental Evaluation of SMaWS

6.1 Overview

This chapter looks at the estimate of overhead caused by monitoring Web Service based on the SMaWS framework. This overhead represents the extra costs of memory usage and average response time that occur when running these Web Services. The various experiments that where performs compared the data measured from the Web Service based on SMaWS framework (also known as “SMaWS Web Service”), and the same Web Service not based on SMaWS (also known as “Basic Web Service”). The following questions are answered in this chapter:

- How large is the Java Virtual Machine (JVM) memory usage that is caused by SMaWS framework?
- What are the differences in average response time caused by SMaWS framework on monitored Web Services?

6.2 Testbed Network

The environments use for testing was a local network that emulates an enterprise setting. The testbed network had a 10Mb/sec Ethernet backbone, and the hardwares used in the test environment are four PCs (Personal Computers) with windows operating system install on them. Software technologies used during the experiment includes:

- **Apache JMeter [58]:** It is a testing framework from Apache Software Foundation [59] designed to be used as a load test tools for analyzing and measuring the performance of services applications.
- **Apache Tomcat 4. 1.3 [51]:** It is a J2EE Web container developed by the Apache Software Foundation, and was used for hosting of web application and Web Service applications.
- **MySQL 5.0 [60]:** it is an open source relational database management systems (DBMS).
- **TCPMon 2.0 [61]:** It is an open-source utility for monitoring data flowing on a TCP connection.
- **J2SDK 1.3 [62]:** It is a Java platform, which enables the deployment of Java applications on desktops and server.

Figure 6:1 shows a diagram of the testbed networks used in the experiment. Table 4 provides a description of testbed Network infrastructure for the measurement of the overhead caused by the SMaWS Framework.

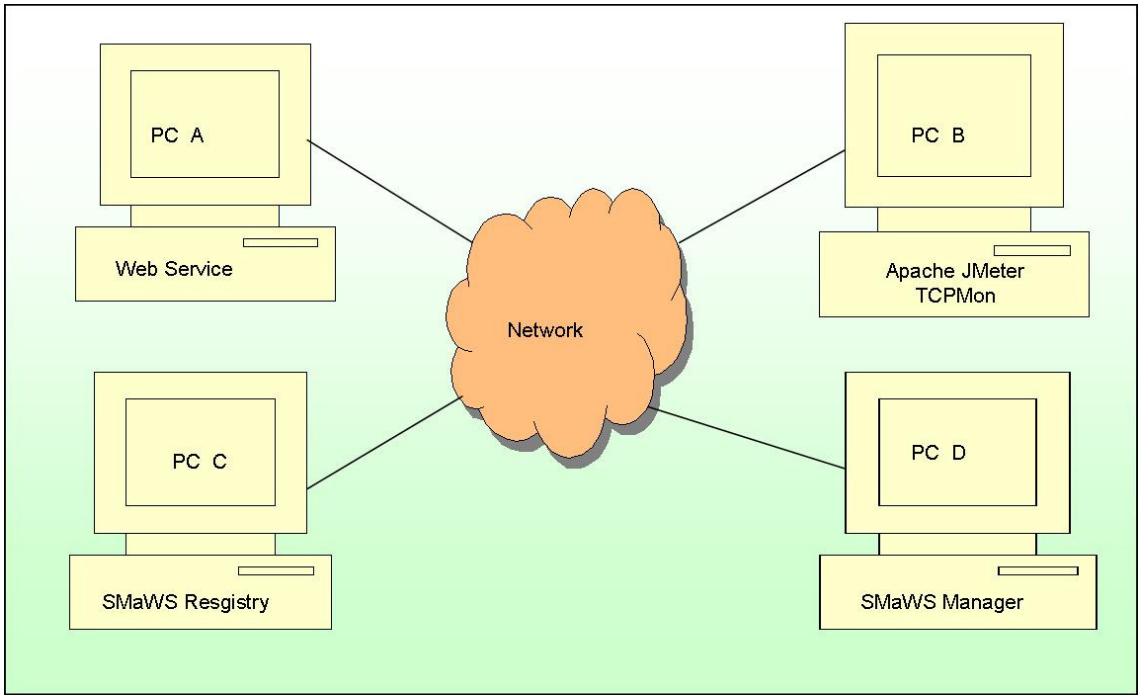

Figure 6:1 Testbed - network for measuring the Overhead caused by monitoring web services based on SMaWS framework

Table 4, Description of Testbed network infrastructure for the measurement of the overhead caused by the SMaWS Framework.

PCs names	**Hardware Properties**	**Software/Application Deployed**	**Description**
PC A	Processor speed: 2 .2 GHz Operating System: Window 2000 Physical memory: 512 MB	Basic Web Service , SMaWS Application, Tomcat, J2SDK	This PC host Web Service applications that are being tested.
PC B	Processor speed: 2 .4 GHz Operating System:	Apache JMeter, TCPMon,	This PC host client applications that generate requests and send to the remote Web services.

	Window XP Physical memory: 512 MB	J2SDK	
PC C	Processor speed : 1.8 GHz Operating System : Window XP Physical memory: 1024 MB	SMaWS Registry, MySQL Server, J2SDK	This PC hosts the SMaWS Registry and database, which is required for the authentication of user. It is also used by the SMaWS Manager application
PC D	Processor speed: 1.7GHz Operating System: Window 2000 Physical memory: 512 MB	SMaWS Manager application, Tomcat, J2SDK	This PC hosts the SMaWS Manager application, which is deployed when the SMaWS Web Service is deployed and tested.

6.3 Experiment, Results and Discussion

The primary propose of this experiment is to compare a "Basic Web Service" and a SMaWS based Web Service, in order to determine the overhead caused by SMaWS framework. During the experiment, a sample Web Service ("Basic Web Service") was implemented that provide the service GetVersion, and another copy of the same sample Web service was also created that has been instrumented with the SMaWS framework ("SMaWS Web Service"). These sample Web Services returns a String when their services GetVersion is invoked by a remote Web Service consumer applications with valid licenses. The following sub-sections examine each of the above-mentioned questions.

6.3.1 How large are the memory usage of SMaWS framework?

The Java API provides three methods in the java.lang.Runtime class: freeMemory, totalMemory, and maxMemory. These methods provide information about the JVM's memory usage. Memory usage was estimated using the differences in total memory and free memory, which was measured through calls to java.lang.Runtime in the Web Service application code.

These measurements were carried out when only one of the Web Services were deployed on the tomcat server. The measurements process was repeats 10 times and the average memory usage was then calculated. Table 5 shows the result of the experiments measuring the additional memory usage caused by the SMaWS framework.

Table 5 Result of the Experiment Measuring the additional Memory Usage caused by SMaWS framework

Measured Value [Units]	Basic Web Service (=A)	SMaWS Web Service (=B)	Difference (B-A)	Relative difference (=(B-A/A)) [%]
JVM Memory Usage [MB]	4,414219	5,320654	0,906435	21%

6.3.2 What is the difference in average response time caused by SMaWS framework on monitored Web Services?

The measurements of the average response time were performed in a local network as mentioned above. During the measurements of the response time, only single copies of the sample Web Service were deployed, so that the effects of other applications running

on the tomcat server are minimized. The measurements include the average response time for an increasing number of Web Service consumer requested.

In this experiment, the measured Web Service response time represents the round-trip time, when a Web service consumer application sends out a request until it gets the response. We use Apache JMeter to measure the response time of the Web Services, as it could be configured to simulate a varied number of simultaneous Web Service consumer applications. The Apache JMeter acts as the Web Service consumer application that generates and sends SOAP messages, which is embedded in the HTTP protocol.

The TCPMon tool was use to capture the Http request send by a Web Service consumer application to a Web Service application. The information read from the captured Http-request data was used to enable the JMeter generate requests sent to the Web Service application.

The measured results are from several thousand requests, which are established to give a better measurement of the actual response time. Typically tests involved 10, 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500 5000 requests. The results presented are averages from each group of requests from run. Finally, the results of the Web Service response time measured by the JMeter are depicted in Figure 6:2.

Table 6 shows the results of the experiment, which estimated the additional Average Response Time caused by SMaWS framework, when the average response time of the total number of requests executed by each Web Service was considered.

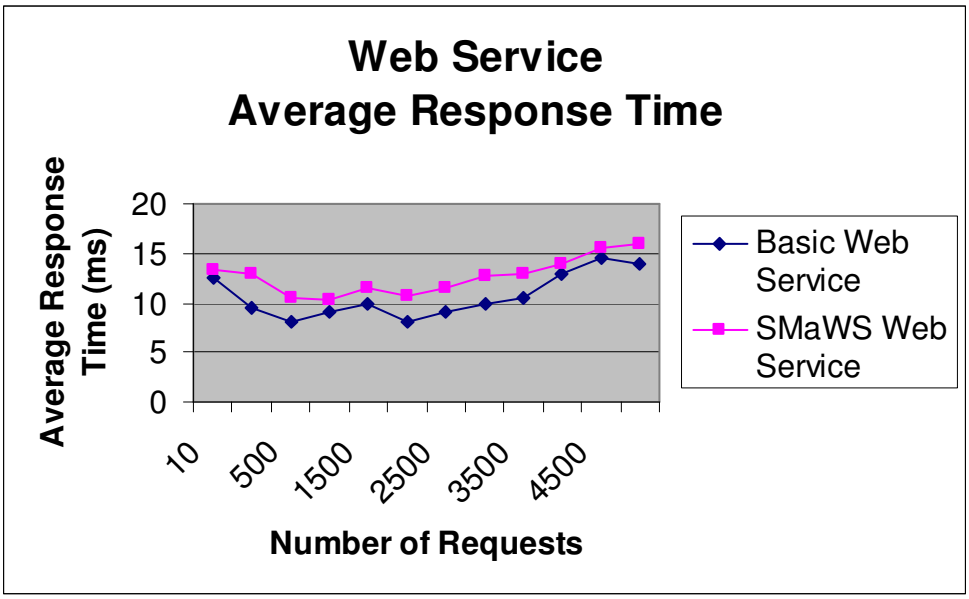

Figure 6:2 Experimental Results of Web Service Average Response time

Table 6 Result of the Experiment measuring the additional Average Response Time caused by SMaWS framework

Measured Value [Units]	Basic Web Service (=A)	Basic Web Service (=B)	Difference (B-A)	Relative difference (=(B-A/A)) [%]
Average Response Time [ms]	10,67	12,66	1,99	19%

6.3.3 Discussion

This section discusses the results obtained from the aforementioned experiments perform above.

Figure 6:2 illustrates the evolution of the average response time for both basic Web Service and SMaWS based Web Service. The graph shows that both average response

time lines follow approximately the same shape, indicating that SMaWS does not induced deviation in the runtime activities of the monitored Web Service.

Table 5 and Table 6 show that SMaWS based Web Service causes an increase of 21% increase the Java virtual Machine (JVM) memory usage, and a 19 % increased in the average Web Service response time respectively. This estimated overhead induced by the SMaWS framework on the monitored Web Service Application is acceptable to many customers. Because, when Web services are deployed over the internet, the network delay is higher, and the relative increase in average response time caused by the SMaWS is lower.

Chapter 7

Related Works

In this chapter, we shall look as some recent papers, products and standardization works in the domain of Web Service management. Most of the major recent related work on standardization of Web Service Management concentrated on the comprehensive description and differentiation for Web Services. Apache Axis [40] is used as a based infrastructure by most of the related works. However, none of the related works provides means for deeper visibility in to the Web Service runtime activities as addressed in my work. Since the approach I use in the management of Web Service focuses on carrying out the monitoring activities at the Web Service layer that is from within the Web Service.

Overall, the current state of related work in Web Service management concentrate on the management of a Web Service as a whole, in contrast of managing the Web Service internal activities, and in the management of the relation ship or interaction between Web Services.

7.1 Web Service Management Frameworks

7.1.1 Farrel and Kreger Web Service management Approaches

In the white paper from Farrel and Kreger [17], they proposed Web Service management approaches. The paper presents an overview about application management, several principles for the management of Web Services, and management patterns that should allow a Web Service developer balance between management requirements and

complexity implications. We shall concentrate only on several principles proposed for management of Web Service, which includes:

- **Separate the management Interface:** the Web Service management interface should be separated from the business interface, such that the business interface should be published in a public UDDI registry, while the management interface should be publish in a separate private UDDI registry that is only access by the management system as client. The separation of the management interface from the business interface will enable the management operation from interfering with the business operation during the search of services based on the interface content.
- **Data collection by the run-time infrastructure**: the collection of the core metric should be carry out by the Web Service execution environment. Here the Web Service execution is the SOAP run-time on the service provider's host. The SOAP- processor provides managed interfaces that collect execution statistics about the invocation and response of the Web Service, as well as allows the control of the SOAP processor itself.
- **Use an Event collector**: intermediate Web Services that act as event collectors should be the use, to signal the occurrence of significant events from a managed Web Service to a management system. The Event collector Web Service will allow the managed Web Service to signal its events in a manner that is independent of the management system platform and implementation.

An IBM Web Service Management Tool kit [18] is present at the end of the paper that implements the data collection Execution environment. It is based on the Apache Axis SOAP engine [40] and implements a management proxy that create a JMX agent within, which automatically creates MBeans for deployed Web Services. The Web Service Toolkit use these facilities to monitors and collects run-time data from both SOAP server and the deployed Web Services. The tool kit does not provide an event collector as discussed in the paper.

On the other hand, the work carried out in this paper does not address dependence among Web Services. This Work [17] encourages Web Service hosting environment to perform Web Service monitoring and data collection. SMaWS framework in contrary supports the act of monitoring of the Web Service at the Web Service layer. The prototype IBM Web Service Management Tools kit collects only a limited variety of data for predefined QoS metrics and therefore provides limited visibility into the Web Service activity as compare to SMaWS Solution.

7.1.2 Web Service Level Agreement (WSLA) Framework

The Web Service Level Agreement (WSLA) framework [2, 4, 19, 20] was developed by the International Business Machines (IBM) Corporation for specifying, creating and monitoring of the Service Level Agreement (SLA) for Web Services. An SLA defines the agreement between a service provider and a service recipient for a level of performance for a particular service. The framework allows service providers and their customers to define the QoS service aspects of a service and their Web Services. The WSLA framework consists of Web Service Level Agreement (WSLA) language to specify the SLAs and a runtime architecture that consists of several SLA monitoring services, which may also be delegated to others independent third parties, to ensure any evaluation is objective and accurate. The WSLA language offers flexibility on how to describe a service, which make WSLA applicable to any inter-domain management scenarios, despite the fact that it was primarily designed for the management of Web Services.

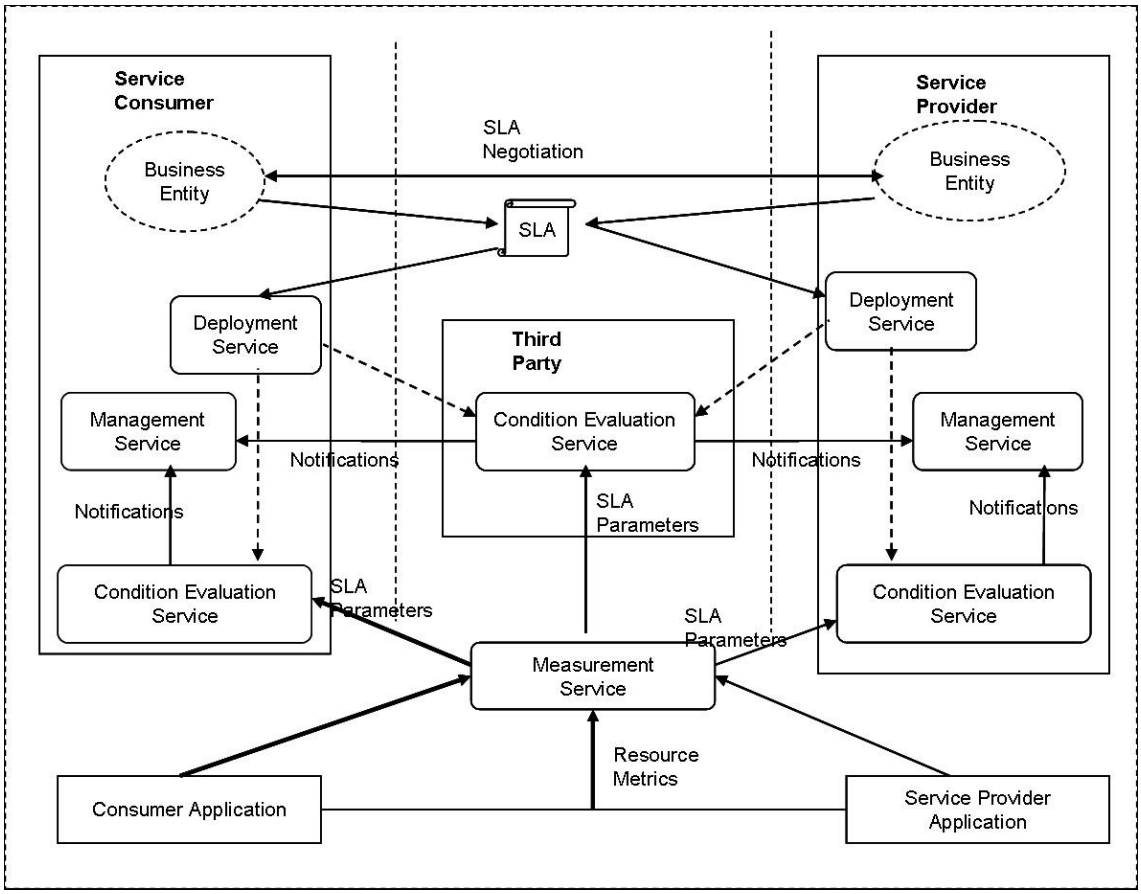

Figure 7:1 WSLA Framework

WSLA Monitoring Services

These SLA monitoring services includes Establishment Service, Deployment Service, Measurement Service, Condition Service, and Management Service. Figure 24 illustrates the monitoring services involved in the WSLA framework when multiple parties are involved. Services that may be outsourced to third parties are either measurement services or condition evaluation services.

The **Establishment Service** consists of SLA negotiation process that is performed either by humans or by Business Entities from the contractual parties as mention in [21]. The process involves the service consumer retrieving the WSLA template with metrics offered by the provider, and filling in his defined SLA parameters with Service level Objectives

(SLO), that is, thresholds according to his business goals and later submitting it to the service provider for approval. Examples of this SLA parameters are response time, throughput etc. If the service provider accepted the SLAs, then both the service provider and consumer can delegate to a management third party the task to partly or fully measure and monitor the SLAs, if there exist no trust between the contractual parties. After the SLAs have been finalized, and third party is defined, both the service consumer and provider then generate the document in the Service Deployment Information format, which is use for deployment.

The **Deployment Service** validates the created SLA and ensures that the third party monitors only SLAs assigned to them.

The **Measurement Service** maintains information about the current system configuration, and run-time values of measured metrics that are part of the SLA.

The **Condition Evaluation Service** is responsible to constantly monitor (that is, by comparing) the measured or calculated values of SLA parameters from the Measurement Service with Service Level Objectives (SLOs), and notifies Management Service of the contractual parties. They can be inside the provider, the consumer, or management third parties (probes or intermediaries).

The **Management Service** is responsible to receives notifications send by the Condition Evaluation Service, and take appropriate corrective actions on behalf of the managed environment if a SLO were violated. The Management Service consists of several port types with operations for the exchange of management information (such as: *GetMetricValue, ParameterUpdate, GetSLAParameterValue, Notification* and *StandardMeasurementIF*) and can be implemented as a part of an existing systems management platform. The Management Service is inside the service provider, and consumer management environment.

WSLA language

The WSLA language specification [22] is the core of the WSLA framework, and is a XML based language for specifying SLAs related information. It specifies how the electronic contract (SLA) among the service consumer, service provider, and third parties, as well as the way interaction among them be carried out. The explicit representation of service level objectives and action guarantees provides a very flexible mechanism to define obligations on a case-by-case basis. The WSLA is designed with the following parts:

- **Parties**: This part identifies all the contractual parties and their technical properties such as their interface definition (for example the port for receiving notification). It consists of subpart **Signatory parties** that is use to define the service consumer and provider, and **Supporting parties** that is use to define the third party.
- **Service description:** This WSLA language part defines various SLA parameters and their measurement or computational procedure for **Service Objects**. A "Service Object" defines an abstraction of all conceptual elements (e.g. WSDL Operation, Binding) of a service. SLA parameters and the corresponding metrics represent the monitored properties of a service object. An example of a SLA parameter is the mean of response time. The **Measurement Directive** or **Function** subpart Information defines how the metrics are measured or computed. The Services description information is used by the Measurement Service to know how to measure and compute aggregate (e.g. mean, median, minimum, maximum of response time) metrics from resource metrics.
- **Obligations**: This part of the WSLA language defines the guarantees, and constraints that may be imposed on SLA parameters. The WSLA language provides two type of obligations that is: Service Level Objectives (e.g. mean response time < 2 s), and Action guarantees (e.g. if SLO is violated,

send notification). This information is used by the Condition Evaluation Service to determine if a SLO has been violated.

SLA Compliance Monitor Implementation

The WSDL Framework contains several tools for creation, deployment, and monitoring of SLA. The Web Service Authoring toolkit [23] supports the creation of WSLA documents and templates, and the filling of those templates at subscription time. This tool is provided as a Service WSLA Authoring Service in the **SLA Compliance Monitor**, which is publicly available as part of the IBM Emerging Technologies Tool Kit (ETTK) version 1.0 [18], previously known as the IBM Web Service Tool Kit (WSTK).

The SLA Compliance monitor is Java based, and consists of a general-purpose Measurement service, Condition Evaluation service, Deployment service, and a WSLA Authoring Service. The implemented general-purpose Measurement service provides metric definitions with a set of functions, and includes multiple data providers (plug-ins) that are use to interpret and execute measurement directives to read measurement data. For example, the JMX based metering service in IBM Web Services Toolkit (WSTK).

The implemented Condition Evaluation Service provides a wide range of logical predicates, which are use for comparing the values of SLA parameters from the Measurement Service with thresholds defined in the Service Level Objectives (SLOs).

The implemented Deployment Service analysis and disintegrate the WSLA documents into relevant parts for given Measurement Services and Conditional Evaluation Services. It also manages the life cycle of SLAs via the used of a Simple WSLA repository, and functions, which it provides.

All these SLA monitoring service (Measurement Service, Conditional Evaluation Service, Deployment Service and Authoring Service) are implemented as Web Services and they offer Web Services interface for the exchange metric values among them during run time.

I think my work can be used as a complementary tool to monitor these Web Services. Since the WSLA focuses on management of Service Level Agreement between the enterprises, and my work focuses on the monitoring and management of Web Service operation within enterprises. The WSLA infrastructure has disadvantage of complexity since the design of both Management service and Business Entity performs very complex tasks, which required considerable infrastructure support.

7.1.3 Web Service Offering Language (WSOL) Framework

The Web Service Offering Language (WSOL) framework [3, 24-27] was presented by Vladimir Tosic in his Ph.D. dissertation at the department of Systems and Computer Engineering, Carleton University, Ottawa Canada. In his dissertation [3], he shows why the specification, monitoring and dynamic (run-time) manipulation of classes of service (service offerings) are useful for Web Services and how they are achieved using the Web Service Offering Language, the Web Service Offerings Infrastructure (WSOI), and the developed algorithms and protocols.

Classes of service are a mechanism for discrete variations of service and quality of service (QoS) guarantees. A Web Service provider can offer a limited number of classes of service, from which the Web Service consumer chooses one of them. For example, a bank offers a loan service, which can be differentiated into a long-term loan service and a short-term loan service with different interest rate, and bank customers could choose between them. A class of service of a Web Service is formally described as a Service Offering. A service offering contains formal representations of various constraints (functional, QoS, access rights) and management statements (prices, penalties) that determine the corresponding class of service. Service offering represents a non custom-made contract for anonymous consumer and both the service provider and the customer do not negotiate on the contents of a service offering, instead they negotiate on which service offering the customer consume. Both the service consumer and provider can renegotiate during run-time for a different service offerings, a process called switching or dynamic adaptation.

WSOL also provides a mechanism for the formal representation of the relationships between different service offerings, which is used to achieve dynamic adaptation of a Web Service Composition. In a Web Service Composition the developed algorithms and protocols is used to provide dynamic manipulation of classes of service by dynamically modifying the QoS levels offered by a service without breaking an existing relationship between a provider Web Service and its consumer. It is also use to provide switching, deactivation, reactivation, deletion, and creation of service offerings.

Web Service Offering Language (WSOL)

The Web Service Offering Language (WSOL) [25] is an XML-based language for the comprehensive description of Web Services with services offering. Describing a Web Service in WSOL enables monitoring, metering, and management of Web Services and their compositions. WSOL reuse information (for the specification of functionality, access methods, and location of Web Services) already define in standard Web Service Description Language (WSDL), which make it compatible with WSDL, and only provides additional specification for various constraints, management statements, and classes of service for Web Services. A WSOL file can reference one or more WSDL files and being in a different language, instead as a WSDL extension, enables the WSOL service offerings to be deactivated, reactivated, created, or deleted without any modification of the underlying WSDL files. A WSOL file contains the following main categories of constructs:

- **Service offering:** This is the main concept in the WSOL, and is used for the description and differentiation of service and QoS of Web Services. It contains descriptions of various categories of constraints, management statements, and reusability of constructs. The definition of one Service offering can be applied to one or several Web Services (that implement the same port types). The WSOL files can contain zero or more service offerings and the definition of a service offering can extend across several WSOL files. Inside the WSOL Service offering element, the attribute

"*accountingParty*" is use to determine the accounting party (Third party) responsible to monitor and carry out the billing of the specify service offering.

- **Constraints**: Valid or expected situations as well as invalid or unexpected situations are described using constraints. The WSOL constraint element describe the valid or expected situation related to the provisioning and consumption of Web Services in the way that can be use for automated monitoring [3]. It enables the formally specification of functional (behavioural) constraints, QoS (non-functional, extra functional) constraints, and access rights. In WSOL, every constraint contains a Boolean expression that formally states the condition to be evaluated.
- **Management statements:** are use to specify some categories of management information, such as prices and management responsibilities. A WSOL **statement** is any construct, other than a constraint, that contains important management information about the represented class of service. The concept of management statement provides easier differentiation and better reusability, as it enable different penalties to be easily assign to the same constraint in different service offerings when violated, instead of using an attribute.
- **Reusability elements:** WSOL contains three categories of the reusability constructs, that is, definition of Service Offering, reusability elements and reusability attributes. These reusability constructs are use to enable easier specification of new service offering from existing service offering of the same Web Service or other Web Services.
- **Service offering dynamic relationship**: The WSOL Service Offering dynamic relationship constructs is use to specify what is the appropriate replacement service offering, if a particular group of constraints condition from some other services was not met. WSOL files can contain zero or more service offering dynamic relationships.

In the Web Service Offering Language management framework, the role of the WSOL is to enable the formal specification of various types of constraint and management statement in a format that can be used for automatic generation of constraint-checking code. It describes for Web Services what QoS metrics to measure or calculate; what constraints (requirements and guarantees) to evaluate; when, where, and to some extent how to perform these monitoring activities; and what are the monetary consequences of meeting or not meeting the constraints [26].

Web Service Offerings Infrastructure (WSOI)

The Web Service Offerings Infrastructure (WSOI) [26, 27] is designed to enable monitoring and manipulation of service offering. It is an implementation of the WSOL framework, an extension of Apache Axis (Apache eXtension Interaction System) SOAP engine and its implementation is Java-based. WSOI support the following monitoring activities that include measurement and calculation of used QoS metrics, evaluation of WSOL constraints, and accounting of executed operations and evaluated constraints. It uses WSOI-specific handlers and chains to perform these monitoring activities. These Monitoring activities can be performed by the Service Provider, Service Consumer, or third party (SOAP intermediary). On the other hand, the WSOI implements the algorithms and protocols for the dynamic manipulation of service offerings, there by enable the WSOI to provide automated management activities.

The main differences between the WSOL and WSLA as Web Service management framework for differential quality of service (QoS) is the WSLA enables the specification, creation, and management of the custom-made SLAs, while the WSOL language enables the specification, monitoring and management of class of service for Web Service. The classes of service approach are simpler to implement and incurs relatively low run-time overhead compared to the custom-made SLAs approached. For example, in a custom-made SLAs approach, a Web Service provider with 100 customers needs to use 100 custom-made SLAs, while a Web Service provider using the class of service approach, might provide only 5 classes of service for similar numbers of customers to choose

between them. In addition, management of classes of service is generally simpler, faster, and incurs less run-time overhead than custom-made SLAs.

However, WSOL is similar to WSLA in their area of focus that is to offer means for differentiation of QoS Service, and the establishment of SLA between the between enterprises, which can be observed by the Service Provider, Service Consumer, and Third party. I think my work can also be use as a complementary tool to monitor Web Services my work focuses on the monitoring and management of Web Service operation within enterprises.

7.1.4 Web Service Distributed Management (WSDM)

Web Service Distribution Management version 1.0 (WSDM) [1] is the current management standard from the Organization for the Advancement of Structured Information Standard (OASIS) Web Services Distributed Management (WSDM) Technical Committee (TC). The WSDM standard is designed to address management integration problems that arises when enterprises uses a variety of management systems (that co-exist) to manage diverse IT resources and applications from multiple vendors within their IT infrastructure. It provides enterprise with a standardized approach that uses Web Service for distributed management of both Web Services and variety of IT resources. It exposes management interfaces through two specifications: Web Services Distributed Management: Management Using Web Services (MUWS) and Web Services Distributed Management: Management Of Web Services (MOWS) specifications.

The MUWS specification provides a framework that defines how to represent and access the manageability interfaces of resources as Web Services. It defines how to describe the management capabilities of managed resources using WSDL documents, and provides manufacturer of IT resources the ability to expose management interfaces of their resources in a standard way, regardless of how the internal instrumentation is done. In addition, this enhances management applications that support Web Services with the

ability to manage a variety of IT resources with one set of instrumentation, as well as reduce the amount of custom support they might required.

The MOWS specification is an extension to MUWS. It defines how to manage a Web Service as any other IT resource and how to describe and access the managed Web Service using MUWS. The managed Web Service only provides measured values for metrics, which are collected by the external management application. Therefore, the management of the Web Service status, aggregation of measured metrics values, evaluation of conditions, billing, and supporting specific management tasks(e.g., SLA monitoring) are not performed by the managed Web Service or runtime infrastructure environment, instead by the external management applications.

WSDM facilitate communication between management applications and resources across numerous vendors, platforms, and technologies, enable end-to-end information collection, and avoid the use of agent, which are all shortcoming in tradition management solution. On the other hand, the fact that all the management activities are perform by the management application increases the number of exchanged SOAP messages, and thus delays and run-time overheads. WSDM does not provide any support for differential of Service offerings (QoS).

7.1.5 Web Service Management (WSMN)

Web Service Management Network (WSMN Network) [28, 41,42,43] was developed by Hewlett-Packard laboratories and it provides a logical overlay network for federated management of Web Services that interact across different administration domains (e.g.; different businesses). WSMN is designed to manage the relationship of Web Services through the management of Service Level Agreements (SLAs). These SLAs can be explicitly agreed by both the Web Service provider and Web Service consumer or implicitly introduce between Web Services for the purpose of management (“implicit SLAs”).

The WSMN architecture consists of a network of intermediaries (proxies), with each position between a Web Service and the outside world as shown in figure 7.1:2 below. These WSMN intermediaries [41, 42] are used to monitor and enforce Service Level Agreements (SLAs), and to exchange control information. They communicate among each other through a set of management protocols are: life-cycle protocols (i.e.: for the initiation and sustenance of the WSMN), measurement protocols (i.e.: for the exchange of SLA monitoring results), and SLA assurance protocols (i.e.: for runtime optimization and control of SLAs).

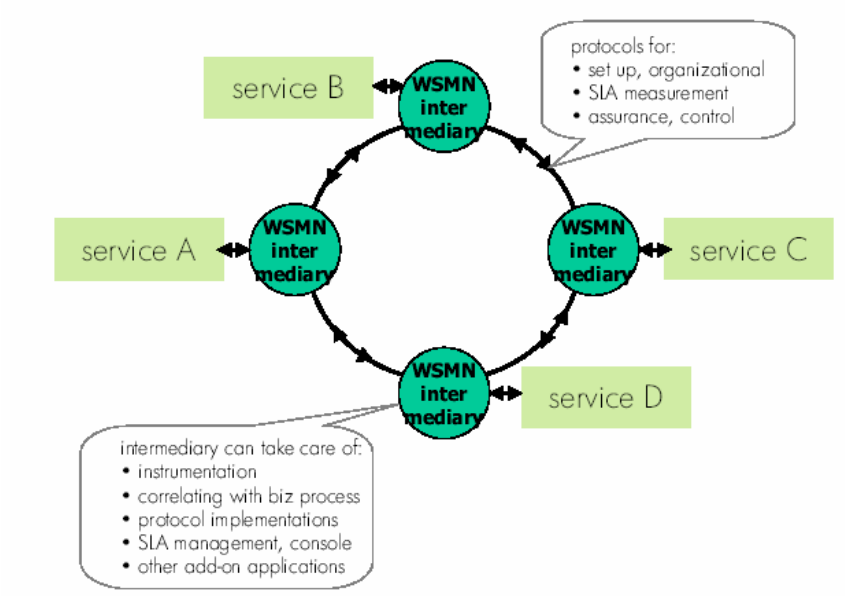

Figure 7:2 Web Service Management Network [28]

Contrary to SMaWS, the WSMN focus on protocols for sharing management information among Services and SLA management in a federated environment.

7.2 Web Service Management Commercial Products

This section covers an overview of some industrial products for Web Service Management. Several of the commercial products have common characteristics of carrying-out the Web Service management activities at the Web Service hosting environment (Application Server) as well as the use of the Web Service technology for the transfer of management information. Overview of some of the major industrial products, which I studied during my PhD research are shown below. It should be noted that significant changes do occur in the market, since new and more efficient products had occur after my study.

Several of the commercial products are based on WSDM, and Web Service based solution (WSDM) uses SOAP protocol for communication between the Manager application and the agent, which result to high consumption of memory, more network bandwidth, and more Cycles than the use of Java RMI as use in the SMaWS framework. On the other hand, the use of SMaWS framework enable the Web Service management activities to be carried-out at the Web Service layer, and the use of Java RMI for the transfer of Web Service management information incurs less run-time overhead than Web Service based solution.

7.2.1 Actional's Web Service Management products

Actional offers a Web Service management platform [32], which is a combination of two separate products: Looking Glass, SOAP station and Ghost Agent. Actional's SOAP station is a Web Services intermediary that act as a management proxy between Web Services and controls Web Service monitoring, security and access control, interoperability, traffic flow and policy implementation. Actional's Ghost Agents are non-intrusive and system-specific agents that are available for several application servers to

actively monitor and alert on service activity. Ghost Agents are weightless and non-intrusive and enable end-to-end visibility. Actional Looking Glass is the centralized management components of Actional's SOA solution that provide an interface to enable administrator discover and monitor both Web Service dependencies and performance issues. The Actional Looking Glass component centralization concept is similar to that use by the SMaWS' Management component. Both Actional's SOAPstation and Ghost Agent are non-intrusive and this result in the limitation of QoS metrics that they monitor and control.

7.2.2 Computer Associates Unicenter ® Web Services Distributed Management

Computer Associates Unicenter ® Web Services Distributed Management (CA WSDM) [35] offers a Web Service Management platform based on WSDM specification. It supports both Web Service Discovery and Web Service Monitoring functions within a Web Service infrastructure.

CA WSDM consists of two architectural components: a set of Observers and a Manager. They Observer monitor SOAP traffic from all Web Services ends and report system management data back to a centralized CA WSDM Manager. The Manager component aggregates and analyzes Web Services performance data captured by one or more CA WSDM Observers. The Manager compares the Web Services' system performance against given thresholds and severity levels, and later provides granular view of the Web Services infrastructure to the administrator.

One of the draw back I found in this architecture is that, the main function of the observer is to extract and forward the raw management data for the Manager to aggregate, this cause low network bandwidth, as the network is busy carrying the raw management data that might not be needed by the administrator. The SMaWS framework addresses this problem by having a SMAWS agent embedded in each Web Service to aggregate the raw management data (e.g. response time of each request) in to useful data (e.g. average

response time) that is forwarded to the manager application only at the Administrator requests.

7.2.3 AmberPoint's SOA Management System

AmberPoint's SOA Management System [34] is one of variety of products offered by the company AmberPoint that address the SOA management. This tools is a non-invasive, agent-based architecture for monitoring and management of SOA systems. Monitoring is performed by the agent, which act as a filter that observe the traffic or do not act as intermediaries between Web Services. AmberPoint's SOA Management System manages performance with service-level objectives.

7.2.4 HP OpenView Service Oriented Architecture (SOA) Manager

HP OpenView Service Oriented Architecture (SOA) Manager [32] is a software products that does offers management solution to a variety of SOA resources. SOA resources here represents Web Services application, and any IT resources (for example: databases, message-oriented middleware, and Web Service containers.), which management interfaces are described using Web Services. It enable a centralize monitoring of SOA resources and consists of a set of core components that are distributed in an IT environment. These core components are:

- **Network Services Server** – A central management server that aggregates and presents the raw management data collected from distributed WSM Agents in a meaningful context to the administrator.

- **WSM Agents** – Web Services Management (WSM) agents represent the monitoring module that is integrated within the Web Service container. It collects management data that are forward to Network Service server.
- **WSM Broker** – A proxy server process that act as a Web Service intermediary and provides management capabilities for Web Services containers and their hosted Web Services. The Broker is a separate process from the Web Services container that provides manageability by using broker service (proxies) which is created for each Web Service being managed.

The HP OpenView Service Oriented Architecture (SOA) Manager is based on the WSDM architecture, and it draws back are they result in low network bandwidth. In addition, communications between the various core components are based on SOAP, which results to high overhead as compare to the SMaWS Framework that use Java RMI.

Chapter 8

Conclusion and Future works

8.1 Discussion

As Web Services move beyond the pilot stage to support mission-critical business processes, increasing number enterprises turn to deploy Web Services on which numerous enterprise applications are interdependent. This trend is do to the ability of Web Service technology to enable enterprises integrate their applications at a reduce cost, with greater flexibility and ease. However, despite the fact that Web Service enables enterprises to improve their operational efficiency, it also introduces several new challenges such as dependency among application, and the potential risk for the whole enterprise software system, since a failure in one application can lead to a failure in other dependent applications

This dissertation has addressed interdependence problems among enterprise applications based on Web Services by presenting a SMaWS (Secure Management of Web Services) infrastructure for secure monitoring and management of Web Services. SMaWS enables an earlier detection of poor performance problem in each interdependent Web Service, which would lead to a faster diagnose and fixing of possible performance issue, and thus maximize availability.

The SMaWS framework was presented that enhances the development of Web Service with management capabilities that enables deeper visibility into the Web Service runtime activities, that is, access to Web service identity, reliability, availability, accessibility, and performance information, as well as usage and license information used by Web service consumer applications to access the services of a given Web Service.

A SMaWS Repository and Security concepts has been proposed that enables Web Services to publish Web Service management information, as well as enable only the authorized management application to determine their location. It also ensures both the integrity and confidentiality of parties involved.

All in all, a prototype implementation of the SMaWS Manager Application and Sample Web Service applications are presented, and the experimental results obtained in terms of overhead induced by the SMaWS framework on the monitored Web Service application demonstrated the feasibility of the SMaWS infrastructure.

8.2 Advantage and disadvantages of SMaWS

8.2.1 Advantages

- SMaWS solution provides deeper visibility into Web Service runtime activities as compared to currently Web Service management tools; access to information about the Quality of Service (QoS) of these Web Services; and a unified monitoring environment for Web Services deployed across enterprise business units. It provides administrator with an insight into their Web Services health, as well as means to identify and resolve Web Service performance problems when they occur, thus reducing down time.
- SMaWS solution provides visibility and control into Web Services management activities such as licensing, usage, and testing. Administrator can use the SMaWS Manager application to query information about the license used by the Web Service consumers as well as information about the services they requested.
- The SMaWS framework enables existing and newly develop Web Service applications to be enhanced with management capabilities

8.2.2 Disadvantages

- The instrumentation process used in the framework is Java Management Extension is language specific. This is currently a problem for enterprises, which have deployed their Web Services based on different languages other than Java (that is .NET); these Web Services cannot be monitored by SMaWS infrastructure.
- The SMaWS Security concept depends on public and private key technologies, which are managed by copying and moving key files. Their protection depends on extends of protection of the directories and files on the host systems.

8.3 Summary of Contribution

- This dissertation has described the requirements analysis for monitoring and management of Web Services across an enterprise environment. This is important in order to identify the needs of enterprises.
- This dissertation has presented an infrastructure called SMaWS Infrastructure to enable enterprise securely monitor and manage their Web Services in a unified monitoring environments, and ensures both the SMaWS Manager and managed Web Services applications confidentiality and integrity.
- The uniqueness in this research is in providing a framework (called SMaWS Framework) that enables the instrumentation of existing and new developed Web Service applications, as well as determines Web Service identity, reliability, availability, security, usage, and license used by Web Services consumer to access a given service. This enables the monitoring and management of Web Services application at the Web Service layer, as compare to current Web Service Management products.

8.4 Future works

- Both the SMaWS agent and SMaWS Manager Application need to be extended to have capability to display graphically operational statistics over a year.
- The SMaWS agent can be further developed to provide notification information about violation of SLO (Service Level Objective) by mailing to administrator.
- The most important feature work is how the SMaWS framework be extended to enable Versioning of Web Service that is, how to manage or switch to different version of Web Service. In enterprise scenarios, situation always occurs where upgrade or fixed bugs in their Web Service application need to be done during production and the major problem encounter in Web Service environment is how to transit from the old version to the new version without any negative impact on the Web Service consumer applications.

REFERENCE

[1] "OASIS Web Services Distributed Management (WSDM)" WSDM 1.1 OASIS Standard Specifications, August 2006. On-line at : http://www.oasis-open.org/specs/index.php#wsdmv1.1

[2] H. Ludwig, A. Keller, A. Dan, R.P. King, and R. Franck, "Web Service Level Agreement (WSLA) Language Specification," IBM Corporation, 2003.

[3] V. Tosic, "Service Offerings for XML Web Services and Their Management Applications," in Department of Systems and Computer Engineering, vol. Ph.D. dissertation. Ottawa-Canada: Carleton University, 2004.

[4] A. Dan, D. Davis, R. Kearney, A. Keller, R . King, D. Kuebler, H. Ludwig, M. Polan, M. Spreitzer, and A. Youssef, "Web Services on Demand: WSLA-driven automated management," IBM Systems Journal, vol. 43, pp. 136 – 158, 2004.

[5] T. Mehta, "Adaptive Web Services Management Solutions," Enterprisse Network & Server, vol. 17 2003.

[6] M. B. Juric, S. j. Basha, R. Leander, and R. Nagappan, "Proffessional J2EE EAI", ISBN: 1-861005-44-X, Wrox Press Ltd, December 2001.

[7] D. Watt, "E-business Implementation: A guide to web services, EAI, BPI, e-commerce, content management, portals, and supporting technologies", Chapter 6, "Internal Enterprise Application Integration," pp. 112-139, ISBN: 0-7506-5751-0, Computer Weekly Professional Series, 2001

[8] K. Gottschalk, S. Graham, H. Kreger, and J. Snell, "Introduction to Web Services architecture," IBM SYSTEMS JOURNAL vol. 41, 2002.

[9] P. Cauldwell, R. Chawla, V. Chopra, G. Damschen, C. Dix, T. Hong, F. Norton, U. Ogbuji, G. Olander, M. A. Richman, K. Saunders, and Z. Zaev, "Professional XML Web Services", Wrox Press Ltd, ISBN: 1-861005-09-1, 2001.

[10] E. Cerami, "Web Services Essentials: Distributed Application with XML-RPC, SOAP, UDDI, & WSDL", First Edition, O'Reilly, ISDN: 0-596-00224-6, February 2002.

[11] O. Zimmermann , M. R. Tomlinson , and S. Peuser, "Perspectives on Web Services: Applying SOAP, WSDL and UDDI to Real-World Projects", Springer Professional Computing, ISBN: 3-540-00914-0, 2003

[12] L. Orans and L. E. Pultz, “High-Availability Networking: Toward Zero Downtime”, Gartner Strategic Analysis Report, Gartner Symposium/ITxpo 2001, Lake Buena Vista, Florida, USA, Oct 2001.

[13] H. Hrasna, "J2EE Management Specification - Final Release 1.0", JSR 77, Sun Microsystems, Inc, 19 July, 2002. On-line at: http://java.sun.com/j2ee/tools/management/

[14] "Java™ Management Extensions Instrumentation and Agent Specification", v1.2. Network Circle, Sun Microsystems, Inc., 2002. On-line at: http://rangiroa.essi.fr/cours/langage/00-jmx.pdf

[15] A. Cockburn, "Writing Effective Use Cases", Addison-Wesley, ISBN: 0-201-70225-8, 2001.

[16] I. Jacobson, G. Booch, and J. Rumbaugh, "The Unified Software Development Process", Addison Wesley Longman, ISDN: 0-201-57169-2, 2004.

[17] J. A. Farrell and H. Kreger, "Web Services management approaches," IBM SYSTEMS JOURNAL, vol. 41, pp. 212-227, 2002.

[18] "Emerging Technologies Toolkit version 1.0," IBM alphaWorks, IBM Corporation, 2003. On-line at: http://www.alphaworks.ibm.com/tech/ettkws

[19] H. Ludwig, A. Keller, A. Dan, R.P. King, and R. Franck, "Web Service Level Agreement (WSLA) Language Specification", IBM Corporation, 2003. On-line at: http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf

[20] A. Keller and H. Ludwig, "The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services.," Journal of Network and Systems Management, vol. 11, 2003.

[21] M. Debusmann and A. Keller, "SLA-driven Management of Distributed Systems using the Common Information Model", presented at Proc. of the 8th Int. IFIP/IEEE Symposium, Colorado Springs, USA, 2003.

[22] A. Keller and H. Ludwig, "The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services", IBM Corporation, journal of Network and Systems Management, Vol. 11, No. 1, 2003.

[23] F. Leymann, D. Roller, and M.-T. Schmidt, "Web services and business process management", IBM System Journal, Volume 41, Number 2, 2002.

[24] V. Tosic, B. Pagurek, B. Esfandiari, K. Patel, and W. Ma, "Web Service Offerings Language (WSOL) and Web Service Composition Management (WSCM)," presented at Proceeding of the OOWS'02 (Object-Oriented Web Services), Seattle, USA,, 2002.

[25] V. Tosic, K. Patel, and B. Pagurek, "WSOL – A Language for the Formal Specification of Classes of Service for Web Services," presented at Proceeding of ICWS'03, Las Vegas, USA, 2003.

[26] V. Tosic, Wei Ma, B. Pagurek, and B. Esfandiari, "Web Service Offerings Infrastructure (WSOI) – A Management Infrastructure for XML Web

Services," presented at Proceeding of NOMS (IEEE/IFIP Network Operations and Management Symp.) 2004, Seoul, South Korea, 2004.

[27] W. Ma, V. Tosic, B. Esfandiari, and B. Pagurek, "Extending Apache Axis for Monitoring of Web Service Offerings," presented at Proceedings of the IEEE EEE05 International Workshop on Business Services Networks (BSN'05), Hong Kong, China, 2005.

[28] V. Machiraju, A. Sahai, and A. v. Moorsel, "Web Services Management Network: An Overlay Network for Federated Service Management," Hewlett-Packard Laboratories, 2002.

[29] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal., "Pattern-Oriented Software Architecture, A System of Patterns", Volume 1, John Wiley & Sons Ltd., ISBN: 0-471-95869-7, 1996.

[30] JSR-3 Java Management Extension Specification, Version 2.1, "The Java Community Process," Sun Microsystems, Inc., September 7, 2000. On-line at : http://jcp.org/jsr/detail/3.jsp

[31] JSR 160 Java Management Extension (JMX) Remote API 1.0, "The Java Community Process," Sun Microsystems, Inc., September 7, 2003. On-line at : http://jcp.org/jsr/detail?=160

[32] "HP OpenView Service Oriented Architecture (SOA) Manager 2.11 software, ", Data sheet, Hewlett Packard. On-line at http://h20229.www2.hp.com/products/soa/index.html

[33] "Actional Service Oriented Architecture (SOA) Software Management products", Data sheet, "SOA Management solution", Progress Software Corporation. On-line at: http://www.actional.com/products/docs/brochure_soa_management.pdf

[34] "AmberPoint Service Oriented Architecture (SOA) Management Systems", AmberPoint Inc.. On-line at : http://www.amberpoint.com/solutions/sms.shtml

[35] "Unicenter Web Services Distributed Management (Unicenter WSDM)", Management Solution for Service-oriented architecture (SOA) environment, Computer Associate Inc. On-line at: , http://www3.ca.com/technologies/subsolution.asp?id=3339

[36] D. Alur, D. Malks, and J. Crupi, "Core J2EE Patterns: Best Practices and Design Strategies", Second Edition, Sun Microsystems Press, A Prentice Hall Title, ISBN: 0-13-142256-4, 2003

[37] Mark Roth, "Java Server Pages™ Specification Version 2.0", JSR-000152, Sun Microsystems, Santa Clara, CA, November 24, 2003. On-line at: http://jcp.org/aboutJava/communityprocess/final/jsr152/

[38] Gregory Murray, "Java™ Servlet Specification Version 2.4", Sun Microsystems, Santa Clara, CA, November 24, 2003. On-line at: http://jcp.org/aboutJava/communityprocess/final/jsr154/index.html

[39] R. B. Bloom, "Apache Server 2.0: The Complete Reference ", McGraw-Hill, ISBN: 0-07-222344-8, 2002

[40] "Apache Axis version 1.2", Apache Software Foundation, May 04, 2005, http://ws.apache.org/axis/

[41] A. Sahai, V. Machiraju, M. Sayal, L. Jin, F. Casati, Automated SLA Monitoring for Web Services. In Proc. of the 13th IFIP/IEEE Int. Wsh. on Distributed Systems: Operations and Management, DSOM 2002 (Montreal, Canada, Oct. 2002)

[42] S. Akhil, A. Durante and V. Machiraju: Towards Automated SLA Management for Web Services. Technical Report, Hewlett-Packard Labs, Juli 2002, HPL-

2002-310

[43] S. Akhil, S. Graupner and W. KIM: The Unfolding of the Web Services Paradigm, Technical Report, HP Labs, 2002, HPL-2002-130

[44] "JSR 122: J2EETM Connector Architecture, Version 1.5,", The Java Community Process, Sun Microsystems Inc., 2003. On-line at: http://www.jcp.org/en/jsr/detail?id=112

[45] H. Kreger, W. Harold, and L. Williamson, "Java and JMX: Building Manageable Systems", Addison-Wesley, ISBN: 0-672-32408-3, 2003

[46] "Java Cryptography Extension (JCE), version 1.2.2", Sun Microsystems, 2005. On-line at: http://java.sun.com/products/jce/

[47] "Java Authentication and Authorization Service (JAAS), version 1.0", Sun Microsystems. On-line at: http://java.sun.com/products/jaas/

[48] B. Clifford Neuman and Theodore Ts'o. Kerberos: An authentication service for computer networks, IEEE Communications Magazine, pages 33–38, September 1994

[49] W. Grosso, "Java RMI", Remote Method Invocation, O'Rielly, First Edition, ISBN: 1- 56592 - 452 - 5. October 2001

[50] E. Pitt and K. McNiff, "Java(TM).RMI: The Remote Method Invocation Guide", Addison-Wesley Professional; First Edition, ISBN: 978-0201700435 July 2001,

[51] "Apache Tomcat", Apache Software Foundation, http://jakarta.apache.org/tomcat/.

[52] J. Pritchard, "COM and CORBA(R) Side by Side: Architectures, Strategies, and Implementations" Addison-Wesley Professional; First Edition, ISBN: 978-

0201379457, July 1999

[53] N.A.B. Gray, Comparison of Web Services, Java-RMI, and CORBA service Implementations, School of Information Technology & Computer Science, University of Wollongong

[54] A. K. Ahamed, “Understanding Java RMI Internals”, 2005. On-line at: http://www.developer.com/java/ent/article.php/10933_3455311_2

[55] E. Rescorla, “SSL, TLS: Designing, and Building Secure Systems” Addison-Wesley Professional; First Edition, ISBN: 0-201-61598-3, October 2000.

[56] Bouncy Castle, Open source Java Cryptography Extension (JCE) Provider. On-line at: http://www.bouncycastle.org

[57] D. Hook, “Beginning Cryptography with Java”, Wrox, chapter 4 “The RSA Algorithm”, pp.85 – 93, ISBN: 0-764-59633-0, August 19, 2005

[58] “Apache JMeter”, Apache Software Foundation Online at: http://jakarta.apache.org/jmeter/

[59] Apache Software Foundation, http://www.apache.org/

[60] MySQL, Open source Database Management Systems (DBMS). Online at: http://www.mysql.com/products/enterprise/server.html

[61] “TCPMon: An open-source utility to Monitor A TCP Connection”,Java.net community project, Online at: https://tcpmon.dev.java.net/

[62] Java 2 Standard Edition (J2SDK), Sun Microsystems Inc.. On-line at: http://java.sun.com/javase/index.jsp

[63] M. Crouch, “Enterprise Application Integration Shaping the Reality of Today”, Managed Business Consulting Inc. On-line -article, January 10, 2003. On-line

at: http://www.mbconsulting-inc.com/Downloads/EAIToday.pdf

[64] D. Serain, and I. Craig, “Middleware and Enterprise Application Integration”, Springer; First Edition Chapter 7, “Message-based Middleware”, pp. 147-161, ISBN: 1-852-33570-X, September 17, 2002

[65] World Wide Web Consortium (W3C), Web services, 2002, http://www.w3c.org/2002/ws/

[66] E. Newcomer “Understanding Web Services: XML, WSDL, SOAP, and UDDI” Addison-Wesley Professional; First Edition, Chapter 4, “Accessing Web Services: SOAP”, ISBN: 0-201-75081-3 May 13, 2002

[67] E. Christensen, Curbera, F., Meredith, and G., Weerawarana,, "Web Services Description Language (WSDL) v1.1.," 2001.

[68] S. Balsamo, A. Di Marco, P. Inverardi, M. Simeoni, Model-Based Performance Prediction in Software Development: A Survey, IEEE Transactions on Software Engineering, Vol. 30, No. 5, MAY 2004, pp. 295-310

[69] S. Kounev and A. Buchmann, Performance Modeling and Evaluation of Large-Scale J2EE Applications, Proceedings of the 29th International Conference of the Computer Measurement Group (CMG) on Resource Management and Performance Evaluation of Enterprise Computing Systems (CMG-2003), Dallas, Texas, December 7-12, 2003