Security Policy Architecture for Web Services Environment

PhD Thesis

Khalid Aldrawiesh

This thesis is submitted in partial fulfilment of the requirements for the

Degree of Doctor of Philosophy

Software Technology Research Laboratory (STRL) Faculty of Technology,

De Montfort University

August 2011

Dedication

To

My Mother and Father

To

My Wife and Children

To

My sisters and brothers

For their love and encouragement

During this time of challenges

Declaration

I declare that this work was presented by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degrees or professional qualification.

This work is original work undertaken by me for the degree of Doctor of Philosophy, at the Software Technology Research Laboratory (STRL), Faculty of Technology, at De Montfort University, United Kingdom.

Abstract

An enhanced observer is model that observes behaviour of a service and then automatically reports any changes in the state of the service to evaluator model. The e-observer observes the state of a service to determine whether it conforms to and obeys its intended behaviour or policy rules. E-observer techniques address most problems, govern and provide a proven solution that is re-usable in a similar context. This leads to an organisation and formalisation policy which is the engine of the e-observer model. Policies are used to refer to specific security rules for particular systems. They are derived from the goals of management that describe the desired behaviour of distributed heterogeneous systems and networks. These policies should be defended by security which has become a coherent and crucial issue. Security aims to protect these policies whenever possible. It is the first line of protection for resources or assets against events such as loss of availability, unauthorised access or modification of data. The techniques devised to protect information from intruders are general purpose in nature and, therefore, cannot directly enforce security that has no universal definition, the high degree of assurance of security properties of systems used in security-critical areas, such as business, education and financial, is usually achieved by verification.

In addition, security policies express the protection requirements of a system in a precise and unambiguous form. They describe the requirements and mechanisms for securing the resources and assets between the sharing parties of a business transaction.

However, Service-Oriented Computing (SOC) is a new paradigm of computing that considers "services" as fundamental elements for developing applications/solutions. SOC has many advantages that support IT to improve and increase its capabilities. SOC allows flexibility to be integrated into application development. This allows services to be provided in a highly distributed manner by Web services. Many organisations and enterprises have undertaken developments using SOC. Web services (WSs) are examples of SOC. WSs have become more powerful and sophisticated in recent years and are being used successfully for inter-operable solutions across various networks. The main benefit of web services is that they use machine-to-machine interaction.

This leads initially to explore the "Quality" aspect of the services. Quality of Service (QoS) describes many techniques that prioritise one type of traffic or programme that operates across a network connection. Hence, QoS has rules to determine which requests have priority and uses these rules in order to specify their priority to real-time communications. In addition, these rules can be sophisticated and expressed as policies that constrain the behaviour of these services. The rules (policies) should be addressed and enforced by the security mechanism. Moreover, in SOC and in particular web services, services are black boxes where behaviour may be completely determined by its interaction with other services under confederation system.

Therefore, we propose the design and implementation of the “behaviour of services,” which is constrained by QoS policies. We formulate and implement novel techniques for web service policy-based QoS, which leads to the development of a framework for observing services. These services interact with each other by verifying them in a formal and systematic manner. This framework can be used to specify security policies in a succinct and unambiguous manner; thus, we developed a set of rules that can be applied inductively to verify the set of traces generated by the specification of our model’s policy. These rules could be also used for verifying the functionality of the system.

In order to demonstrate the protection features of information system that is able to specify and concisely describe a set of traces generated, we subsequently consider the design and management of Ponder policy language to express QoS and its associated based on criteria, such as, security. An algorithm was composed for analysing the observations that are constrained by policies, and then a prototype system for demonstrating the observation architecture within the education sector. Finally, an enforcement system was used to successfully deploy the prototype’s infrastructure over Web services in order to define an optimisation model that would capture efficiency requirements.

Therefore, our assumption is, tracing and observing the communication between services and then takes the decision based on their behaviour and history. Hence, the big issue here is how do we ensure that some given security requirements are satisfied and enforced? The scenario here is under confederation system and based on the following:

- System’s components are Web-services.
- These components are black boxes and designed/built by various vendors.
- Topology is highly changeable.

Consequently, the main issues are:

- The proposal, design and development of a prototype of observation system that manages security policy and its associated aspects by evaluating the outcome results via the evaluator model.
- Taming the design complexity of the observation system by leaving considerable degrees of freedom for their structure and behaviour and by bestowing upon them certain characteristics, and to learn and adapt with respect to dynamically changing environments.

Acknowledgements

First and foremost, my deepest thankfulness goes to Almighty ALLAH for all his bounties and blessings, and for giving me the ability to complete this research.

I was very lucky to have **Professor Hussein Zedan**, the director of the STRL, as my adviser. His scientific support, insightful comments, ideas, visionary approach, support and guidance have greatly helped and supported me in the development my thinking and technical writing, and widening my horizons.

My thankful goes to the first supervisor **Dr. Amelia Platt** for her support, management and guidance during my studying. Also, my thankful goes my great second supervisor **Dr. Francois Siewe** through very exciting and enjoyable discussions and his patience. His scientific and moral support through the difficulties I faced during my research was crucial to my success. Also, thankful goes to **Dr. Helge Janicke** for his support at the beginning phase.

Express thankful must go to **Professor. Hongji Yang** for his support and cooperation with during studying, also, great thankful should go to all member of Software Technology Research Laboratory (STRL) especially Mrs. Lindsey Trent and Mrs. Lynn Rayn for their support and management since started in 2005.

I would like to thank all of my colleagues, especially Dr. A. Alqahtani, Dr. A. Al-Ajlan, Dr. A. Al-bayati, Dr. N. Al-alwan, Dr. M. Muqabla, Dr. M. Al-Sammaraei, Dr. O. Al-Shathri, Dr. F. Al-Shathri, Dr. O. Aldabbas, H. Aldabbas, Dr. A. Bajahzair, Dr. Y. Al-Saway, Dr.A. Alhussain, Dr. A. Al-hammad and Dr. M. Sarabb, in STRL at De Montfort University for their valuable suggestions and discussions during the study.

I would also like to thank the staff in the Research Office at De Montfort University for their outstanding management. I am also deeply indebted to my friends Reyad Al-Waddan, Abdulmalek Al-Duhami and Abdullah Al-Asem, for their concern and encouragement through all my study years. As well as, special thankful goes to Dr. A. Murali Rao for his value advice at the beginning stage.

My thankful should go to my sponsor the MOE and member staffs for their financial support, patience and dealing with my requests to issue and extend my scholarship especially the Centre of IT & Computer Department at the MOE Dr. J. Al-ghamdi-Direct manager- and the former managers, Eng A. Al-rawaf, Dr. M. Al-humaid.

Thankful again must go to Minister of Education Prince Faisel Bin Abdullah Al-Saud, Vice Minister Faisel Bin Moamer also, the former Minister of the MOE Dr. Saleh Al-Obaid and Vice Minister Prince Khalid Bin Abdullah Al-Meshari

My special gratitude is due to my dearest brothers and to my lovely sisters and their children and families for their loving support, concern and encouragement. I would like also to express my deepest gratitude to my loving mother and father, who gave their love, pray and support, for everything they sacrificed in their life for me. Without their loving care, encouragement and support, it would have been very difficult for me to achieve my goals.

I would like to express my deepest love and gratitude for my wife, who stood by me in all these difficult years and has offered me her unconditional constant support, patience, encouragement, love and life. Finally, my love goes to my children Abdullah, Shahd, Shaden, Shouq and Shima for the hope and encouragement they have given to me, without knowing it, to complete this work.

Publications:

Throughout the course of the incremental study and research, the results have been reported and published in scientific papers:

1. K. Aldrawiesh, H. Janicke and H. Zedan. Policy-based QoS approach in Web environment, the 1st Saudi Innovation International conference, in proceedings of ICT Workshop, Newcastle University, SIIC2007, 12-13 May 2007, Newcastle, UK.

2. K. Aldrawiesh, F. Siewe and H. Zedan. Security Policy Architecture for Web Services Environment, the 2nd Saudi Innovation International conference, in proceedings of ICT Workshop in Leeds University, SIIC2008, 4-5 Jun 2008, Leeds, UK.

3. K. Aldrawiesh and F. Siewe. QoS Approach in WS-Security Policy Environment, the 3rd Saudi International conference, in proceedings of ICT Workshop in Surry University SIC2009, 5-6 Jun 2009, Surry, UK.

4. K. Aldrawiesh and F. Siewe. An Observable model for developing Security Policy approach in Web services, the 4th Saudi International conference, in proceedings of ICT Workshop in Manchester University, 30-31 July SIC2010, Manchester UK.

5. K. Aldrawiesh, F. Siewe and H. Zedan. An Enhanced Observer Model to Detect Security violations in Web services, in proceedings of the ACM International conference on Intelligent Semantic Web-Services and Applications (ISWSA 2011) April 18-20, 2011, Amman, Jordan. 978-1-4503-0474-0/04/2011.

6. K. Aldrawiesh, Amelia Platt and F. Siewe. Towards Development a Policy-Based Technique for Enforcing Security Violations, the 5th Saudi International Conference, in proceedings of ICT Workshop in The University of Warwick, Coventry, 23-26 June SIC2011, Coventry UK, ISBN:978-0-9569045

List of Acronyms

ACP	Access Control Policy
AGG	Attributed Graph Grammar
API	Application Programming Interface
BEEP	Blocks Extensible Exchange Protocol
COBOL	COmmon Business-Oriented Language
DS	Distributed Systems
UML	Uniform Modelling Language
FSM	Finite State Machine
FTP	File Transfer Protocol
GUI	Graphical User Interface
GT	Graph Transformations
JFLAP	Java Formal Languages and Automata Package
LS	Legacy Systems
LIS	Legacy Information System
IT	Information Technology
HTTP	Hypertext Transfer Protocol
J2EE	Java Platform, Enterprise Edition
LDAP	Lightweight Directory Access Protocol
ODBC	Open Database Connectivity
OOP	Object-Oriented Programming
SOAP	Simple Object Access Protocol
SLAs	Service Level Agreements
URL	Uniform Resource Locator
URI	Uniform Resource Identifier
UDDI	Universal Description, Discovery, and Integration
ADSL	Asymmetric Digital Subscriber Line
SMTP	Simple Mail Transfer Protocol
SLM	Service Level Management
SMS	Short Message Service

PL/SQL	Procedural Language/Structured Query Language
SP	Service Provider
SReq	SReq Service Requestor
SReg	SReg Service Registry
KSA	Kingdom of Saudi Arabia
MOE	Ministry of Education
WIS	Web Information Systems
WSPS	Web Services Protocol Stack
WSM	Web Services Management
WSDL	Web Service Description Language
XML	Extensible Markup Language
WAP	Wireless Application Protocol
W3C	World Wide Web Consortium
ASL	Authorisation Specification Language
KAoS	Policy language
LaSCO	Language for Security Constraints on Objects
ISPS	IPsec Security Policy Specification
KAoS	Policy language
Ponder	Ponder Policy language
PDL	Policy Description Language
Rei	Policy language
DAC	Discretionary access control
MAC	Mandatory access control
SOC	Service-Oriented Computing
SOA	Service-Oriented Architecture
RBAC	Role-Based Access Control
QoS	Quality of Service
ACLs	ACL Access Control Lists
JIT	Just-In-Time (JIT)
QoS-A	Quality of Service Architecture (QoS-A)
EPI	Enterprise Application Integration (EAI)

PDAs	Personal Digital Assistants (PDAs)
WSA	Web Services Architecture (WSA)
ESP	External Service Provider (ESP)
WSM	Web Service Management (WSM)
KPIs	key performance indicators
PAP	Policy Administrative Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PR	Policy repository
PIP	Policy Information Point
PE	Policy Enforcer
PM	Policy Manager
RPCs	Remote Procedure Calls
RMI	Remote Method Invocation
IDE	Integrated Development Environment
LGTS	Layered Graph Transformation Systems
JDK	Java Development Kit
SDL	Specification and Description Language
ADL	Architecture description Language
OMT	Object Modelling Technique
RSC	Rational Software Corporation
OOSE	Object-Oriented Software Engineering
COM	Component Object Model
CLR	Common Language Runtime
IIS	Internet Information Services
ORM	Object Role Modelling
SOMF	Service-Oriented Modelling Framework
EEML	Extended Enterprise Modelling Language
XACML	eXtensible Access Control Markup Language

List of Figures:

List of Tables

Table of Content

CHAPTER 1

INTRODUCTION

Objectives

- *To present the background, motivation and objectives of the research*
- *To highlight the research questions, research methodology and original contributions*
- *To explain thesis's outline and structure*

1.1 Background

Having and managing huge information technology resources have become a difficult mission. These resources must be operated by system managers and take heterogeneous systems, different networking technologies and distributed applications with several considerations to be controlled, this growth has become more complicated and powerful and has changed the nature of computing over the last decade. As computer software, systems, services, visionaries and technologies have become cheaper, smaller and more powerful, they have also become more complex. The information revolution has made these technologies visible and tangible. Most of these technologies and systems are connected via networks, so computer systems and their associated architectures have developed quickly and this leads to that the security of these technologies has become a coherent and crucial issue.

For many organisations and enterprises, security becomes a significant requirement. Formal methods are increasingly being used to protect security issues in the field of development of information systems, where confidentiality, integrity and availability of information are paramount. Security is the first line of protection for information or resources stored in a system and should prevent any event that can result in loss of availability, unauthorised access or modification of data.

However, web Services (WSs) are identified as a means of allowing applications to deal and talk with each other using Extensible Markup Language (XML) messages to exchange via the standard web protocol of Hyper Text Transfer Protocol (HTTP) to request web pages from web servers and join it with XML to pass structured information back and forth between computers. WSs are examples of Service Oriented Computing (SOC). SOC offers a way to create a new architecture that gives reflection components trends toward autonomy and heterogeneity. SOC has grown and become a dynamic field for research and improvement and offers many advantages to support information technology (IT), including the enhancement and management of their systems. SOC is an ongoing topic for research and is currently being investigated regarding its promise to share several resources (Aldrawiesh, Siewe et al. 2011).

1.2 Motivation

As the augmentation, the case that the emergence in the use of computers can be regarded as a major scientific and technological accomplishment which has advanced technical and manufacturing processes. Computer technology has changed rapidly in the recent years. Computer technology and services are under the risk so, researchers and developers are being asked to provide enhanced security in order to keep these systems safe.

However, with the advent of the Internet, new types of computation have emerged to leading to Service-Oriented Computing. In this paradigm, the system can be formed / constructed, and dissolved rapidly. In all cases, these systems need to be highly dependable at all times. This means that they must be reliable, available, secure and safe. In our thesis, we will only consider security. The main challenge here is: How to ensure some given security requirements are satisfied and being enforced, given that services are black boxes/legacy/etc and built by many vendors.

1.3 The Objectives of the Research

The objective of this study is how to effectively ascertain results when observing services occur interactively within a web service environment. Also, how to evaluate this interactivity of services? By considering the aforementioned, it need to expose some of its features in order to achieve a successful assessment, a seamless system must be envisaged, considered and protected. A web service environment and its associated components have many characteristics such as the following:

- Connectivity – access to information is available on a global scale
- Flexibility – the service is available at any time and place
- Interactivity – assessment can be immediate and autonomous
- Collaboration – the ability to share many operating systems and platforms

- Availability – maintaining the system within a safe environment against attack or unauthorised access of data

These characteristics can be used as the criteria to evaluate the quality of web services. Thus, the main goal here is to demonstrate and explore novel techniques that can be developed for use in policy-based QoS for Service-Oriented Computing (SOC). In particular, we intend to present a model for SOC where "policy" is the first priority. This model should open the door for collaboration between services that are widely distributed, flexible and effective as follows:

- Manage and formulate a policy language that declares QoS and the criteria issues in terms of security.
- Design an algorithm for analysing observations that are constrained by policies.
- Develop a prototype system for evaluating observation approach within the education sector to simulate the Noor project at The Ministry of Education (MOE) in the Kingdom of Saudi Arabia.
- Present an enforcement system for successfully deploying the infrastructure over Web Services to classify an optimisation model that would capture efficiency requirements.

Consequently, the outcome upon concluding the evaluation process, related work and case study, the suitability of throughput will have been assessed for implementation within a web service and its addendum environment.

1.4 The Research Question

The overall questions investigated by this research are:

How to build a secure system from vulnerable components? Those components may be (web) services?

In order to illustrate these questions, we have assumed a set of research questions that define and address the problems in detail:

- What is the paramount technology for supporting the Observation approach and its associated architecture?
- What is the most suitable platform for using and applying this technology?
- And what are the effects by using and applying Web services to the Observation system?
- What requirements should Web services and the Observation system fulfil?
- Can we apply the Observation approach into the MOE's systems?

Fundamental to these questions is to study the service behaviour. So construction of the behaviour of a service can only be achieved via observing the interaction between services, by demonstrating the process of building the practical applicability with the utilisation of information technologies.

Future chapters will discuss the research question in the technical realm. Nonetheless, other, non-technical issues affect the needs and requirements for technical solutions. Research into these issues helps guide research into the technical areas. A key question is how to quantify risk. The research issue is how to determine the effects of a system's vulnerabilities on its security. A rigorous technique for determining appropriate solutions will be considered and presented.

1.5 Research Methodology

This research analyses and proposes the principles, rules and postulates that could be viewed during the research. This thesis is shaped by the integration of three main important fields in IT: SOC, observation technique and service behaviour. All these fields increasingly play a major role in the strategy for securing a system. Therefore, there is a growing demand for methodologies and technologies that support the use of these fields for different security purposes. The most important challenge for this investigation is to discover a suitable technology to develop an approach to ensure that security requirements are satisfied and enforced.

1.5.1 Choice of methodology

The aim of this thesis is to explain and analyse the interaction process between services by using observation system based on the significance of the web environment. Having this in mind, the focus throughout the thesis will be the field of web services and security policies in order to identify any factors that might affect them.

This study uses pure scientific research methodology and it examines the adoption and dissemination of security policy for web services. Security policies have to be constantly re-defined, updated and maintained to be a valuable technique. It is, therefore, important to develop to capture best practices for the protection of software applications. Moreover, this study traces the roots of web services from their background and related work in the architectural built surroundings to the present conflicting views of its associated environment. The study then presents a critical assessment of the structure of the observation technique for documenting knowledge, including a process for using that knowledge and environment that is involved. This research, in particular the technique, is prompted by the scarcity of resources for organisations that wish to introduce security into their systems.

1.5.2 Choice of theory

It is important to clarify the type of scientific path that is chosen when conducting pure scientific research. The path will facilitate the progress of designing and then answering the research questions.

One type of research involves an evaluation of the observation system that integrated and represents advanced stage in this research. Another type of research involves gathering background material by using books, journals, articles and papers. The research is intended to take the form of a theory-based/research-based experimental investigation model to produce a plan, which will be implemented, tested, evaluated and summarised. It is, consequently, an exploratory and experimental study for building a secure system in this relatively unexplored area. During this process, data will be extracted, providing a basis for comment. In this way, conclusions and opinions drawn from this research can be evaluated.

1.6 Thesis Contributions

The major focus of our research is to ensure that security requirements are satisfied by studying the behaviour of services during interaction communication and to ascertain how effectively these services can be contacted and implemented. Because several vendors provided information, the sheer size of the services investigated led us to look at different issues that could contribute to the research. These contributions constitute the underlying organisation for a comprehensive infrastructure support for web services. The main contribution of this research is to develop a rigorous approach that specifies and verifies the behaviour of services coupled with the aim of simplifying the task of designing and implementing the observation system and their interactive requirements. Hence the main contributions of the research are:

- Propose observation system that increases surveillance by observing the interactive communication between services and then processes and sends the outcomes to the evaluator model and then to the enforcer model.
- Accommodate and tame the design complexity of the observation system by leaving considerable degrees of freedom for their structure and behaviour and by bestowing upon them certain characteristics and to learn and adapt with respect to dynamically changing environments.
- Devise a novel policy-based technique, which supports the observation system for monitoring the services when interacting with each other, and verifying them in a formal and systematic manner. We developed a set of rules by AGG tools, which can be applied inductively to verify the set of traces generated by a specification of the policy. These rules can also be used to verify the functionality of a system.
- Design enforcement architecture that uses a technique to detect any violation activity by addressing the security policy system using a systematic approach that can be leveraged by the model's requirement within the web context.
- Perform the enforcement system to deploy successfully infrastructures as web services in order to define the optimisation model. This model would efficiently capture requirements via addressing the enforcement system and a systematic approach that can be leveraged by the model specifications.

- Develop and implement a prototype system that supervised by the observation system will manage and reduce risk by adopting the protection features of the enforcement system that can specify a method to describe concisely the set of traces generated by the enforcement tool.

There are other important contributions, for example, the knowledge that is embedded in this thesis, which is based on the above contributions such as:

- Propose and design the observation system, which consists of an enhanced observer model, an evaluator model and an enforcer model. The e-observer will monitor the behaviour of service when interaction happens, the outcomes of which are reported to the evaluator model, which will evaluate the outcomes and then send them to enforcer model for detection.
- The technique of the e-observer uses a proactive or precaution system to minimise the risk to the resources.
- The enforcer model uses an independent technique and, therefore, it can be adapted for use with another solution as its dynamicity.

1.7 Thesis Outline and Structure

The thesis is consists of 8 chapters, which are briefly described in outline below.

Chapter 2	Conduct a comprehensive review of SOC, policy based approaches and QoS. This will enable discussion and comments to be made on related background information derived from the literature to include web services, their benefits, architecture with the scenario of Web services. In addition, it describes the Web Services protocol stack with XML, WSDL and UDDI languages.
Chapter 3	Provides an overview of the background information that has influenced the course of security policies. The background information includes a definition of the term “security policy,” access control models, security policy languages, security threats and the goals of security. The chapter also discusses the impact of

security models.

Chapter 4 Describes the design of the architecture of observation, which will manage the surveillance technique to reduce the risk. This chapter is elaborated the observation and its associated parts.

Chapter 5 Formulates a policy-based technique for verification the observation approach and to describe policy specification languages e.g. ponder policy. In addition, the rules that are designed by AGG's tools for simulation and verification are discussed.

Chapter 6 Describes the design and development enforcement architecture that provides a technique for detecting violations, as well as minimising the risk to resources by using a proactive approach.

Chapter 7 Provides a prototype system that developed upon the conduct and implementation of the experiment. The observation system is embedded within this prototype to manage the security.

Chapter 8 Proceeds to analyse and evaluate the observation approach which integrated with the prototype system. We use a benchmark technique to examine its dependability, this evaluation should demonstrate the practical applicability of the proposed approach of by using tools e.g. AGG and FSM with JFLAP to prove and validate the feasibility of this prototype.

Chapter 9 Sums up the work presented in this thesis. The significance of the main findings is provided. The chapter highlights the most important contributions and then discusses methods and directions for possible future studies.

CHAPTER 2

Background and Related Research

Objectives

- *To present an overview of SOC and its associated parts*
- *To illustrate Quality of Service-QoS*
- *To focuses on Web services and their benefits, architecture, scenario, services and the future of Web services.*
- *To describe Web Services Protocol Stack*

2.1 Introduction

Service-oriented computing (SOC) is a new and emerging pattern for distributed computing systems and e-business processing that uses and exploits services as essential elements to permit a wide range of agile networks of collaborating business applications distributed within enterprises and across organisational boundaries(Nabor, Jos et al. 2005; Massimo, Mourad et al. 2006). The ongoing of SOC technologies has encouraged its capability to be adopted by a number of enterprises, governments, financial and travel agencies. In addition, universities and associated institutes in modern countries have recognised that their computing infrastructure can be used to research SOC-based modelling languages, service verification, validation and automated code generation(Tsai and 2006). SOC's feature uses services as important elements for enhancing applications to generate the service model. SOC is based upon SOA and is a method of re-organising software applications and infrastructure into a set of interacting services.

In this chapter, we will conduct an overview of SOC and its associated environment, such as, web services and their use within policy-based quality of service (QoS). In addition, we will examine, analyse and evaluate existing services, especially focusing on the assignment module in order to explore its functionalities and limitations with respect to its design on the behaviour of the service, which is constrained by QoS policies and to discover novel techniques for protecting our system.

2.2 What is a service?

There is no consensus on the definition of a service. In the context of this research, the term "service" is ubiquitous. To some extent; a service can be defined by considering what it is NOT. Services are not the same as the organisation that delivers the service; neither should they be confused with documents, which either delineate the service e.g. leaflets, web pages or are used in transacting the service e.g. application forms.

The vision of a service as something defined and limited by the boundaries of SOAP, Web Service...etc is artificially constraining and blind to the realities of a world, which is increasingly being driven by services over a variety of protocols. This certainly was not the only missive in my notion to blur the conventional wisdom of what constituted a service(Roberts 2002).

Simply, a service (Hans Weigand, Paul Johannesson et al. 2008) is an abstract resource that illustrates an ability to perform tasks with a coherent functionality from the provider entities to the requester entities. A service in order to be used must originate with an agent who acts on behalf of a provider. The provider is either a person or an organisation. Therefore, a service is a thing that fulfils a purpose. Essentially a service is a worker employed to accomplish a specific end aim for a requester. The aim may be small in scope, like retrieving or exchanging information or wider ranging, such as, executing a business process. Most services are mid-ranging in scope, for example, completing a function. The range of a service is referred to as its grain or source or level of granularity(Hans Weigand, Paul Johannesson et al. 2008). Moreover, a service is self-describing, well controlled, has open components or ingredients that uphold fast, cheap and low-cost composition of distributed applications. Services are presented by service provider organisations that obtain the service implementation, supply their service descriptions and provide related technical and business support. The service specifies a contract between the user/client and the operations that could be expected.

The service may be produced and explored using UDDI while is used SOAP for vendor-neutral communications between applications over HTTPs (Goethals 2002; Newcomer 2002; Avik and Amit 2006; Cavanaugh 2006; Jagadeesh, Nandigam et al. 2006). Services have many features that are autonomous, platform-independent computational entities. They can be used in a platform independent way. As well as to its features, they can be specified, published, explored and assembled in dynamic for developing massively distributed, inter-operable and evolvable systems. Services make functions that can range from answering simple requests to executing sophisticated business processes that require peer-to-peer relationships between possible multiple layers of service consumers and providers. Any piece of programme code and any application component deployed on a system can be re-used and transformed into a network-available service. Services reflect a "service-oriented" approach to programming, based atop of the idea of composing applications by discovering and invoking network-available services rather than building new applications or by invoking available applications to accomplish some task(Papazoglou 2003).

Mainly, services are often built in a way that is independent of the context in which they are used. This means that service providers and the consumers are loosely coupled. Thus, service can be based on their descriptions, terms and conditions.

A user of a service usually relies on a contractual agreement with the provider, including what is provided and what comprises the associated QoS, for example, availability, security and other requirement conditions in order to use it(Francisco, Matthew et al. 2002). The service-oriented approach is a design pattern that indicates the creation of automation logic in the form of services; however, it is independent of particular computer programming languages or operating systems-OS. It permits organisations to disclose their core competencies programmatically over the Internet or different types of networks, e.g. cable, UMTS, XDSL and Bluetooth, using standard (XML-based) languages, protocols and implementing a self-describing interface.

As these service technologies proliferate, a world of co-operating services has developed where application elements come together with little effort to form a network of services. This network could loosely couple to create dynamic business processes and agile applications that can span organisations and computing platforms (M. Papazoglou1, P. Traverso et al. 2006). Services keep the promise of moving beyond the simple exchange of information, the dominating mechanism for application integration today, to the notion of accessing, programming and integrating application services that are encapsulated within old and new applications(M. Papazoglou1, P. Traverso et al. 2006). However, in service-oriented computing, services are the main building blocks out of which new applications evolve. As these in-place services grow, which are accessible in a standardised way, composition languages, such as, BPEL are needed to integrate them and subsequently disclose the resulting artefact as a Web service.

In addition, composition and co-ordination, composition middleware and co-ordination middleware are two complementary phases and techniques. The diagram for a service composition is an aspect that is mainly internal to the implementation of the service that composes other Web services, whereas the protocols for service co-ordination are required properties of the external interactions between Web services (Stefan, Rania et al. 2004).

Therefore, the service is available at a specific endpoint in the network and it receives and sends messages and presents behaviour based on its specification. The service has a particular functionality and is offered with appropriate QoS at its endpoint (S. Weerawarana, F. Curbera et al. 2006). Services should be formed to create applications based on the functionality of the services that are available(Gorton and Reiff-Marganiec 2007). Services become more useful when there is some suitable method, abstracted from technological details, to indicate requirements, which are then used by the system to discover and compose services into an executable application that fulfils our aim(Gorton and Reiff-Marganiec 2007).

2.2.1 What kind of thing is a service?

Certainly, not all services are similar and not all services are simple information-oriented requests/replies. Beyond request/reply, a service might be a user, a worker, a monitor, an agent, an aggregator or even a process. A service is a summary resource that has some characteristics viz. a name, a job, number, job tasks, contact information and policy regarding security and service levels. To request a service, a user/consumer sends a message in accordance to the contact information and policies and then should receive a reply. However, the job of a service is limited and dominated to a single distinct business concept, function or process. This feature is referred as the bounds of a service. Finding the correct bounds is a key element in service definition. Based on its job or needs to complete its job, a service may call upon other services. This service-to-service relationship is called collaboration method(David Booth, Hugo Haas et al. 2004; Michelson 2008).

2.2.2 The Service Framework

The recent spurt in the growth of businesses on the web is being transformed rotated into services that are accessible on it. Figure 2.1 below demonstrates and proposes a framework for the service and its basic associated entities. This diagram illustrates some entities that could be used and joined together when a service is processing e.g. request/respond/send/receive a message regarding a system requirement. A brief description of these entities follows.

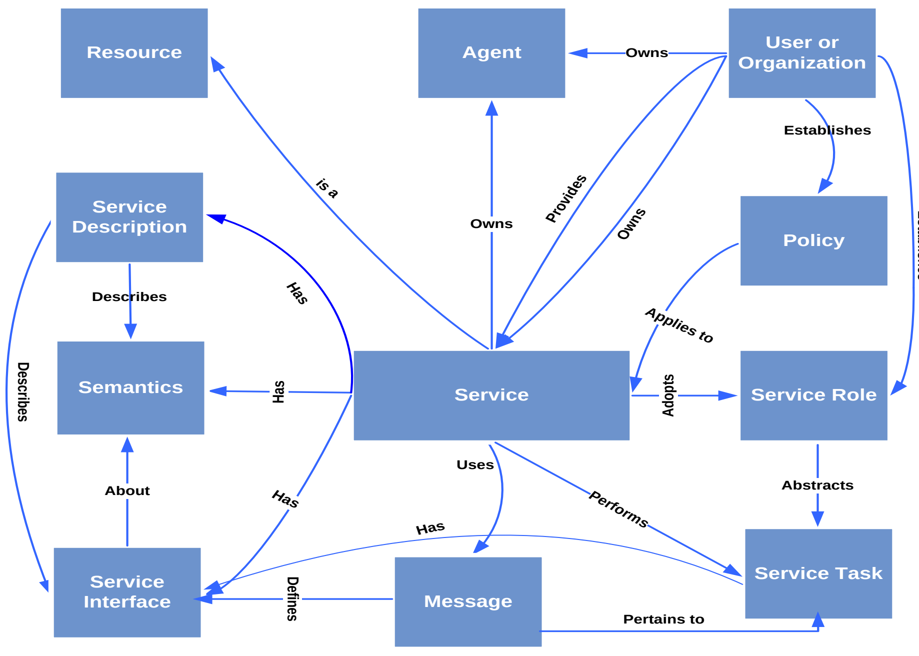
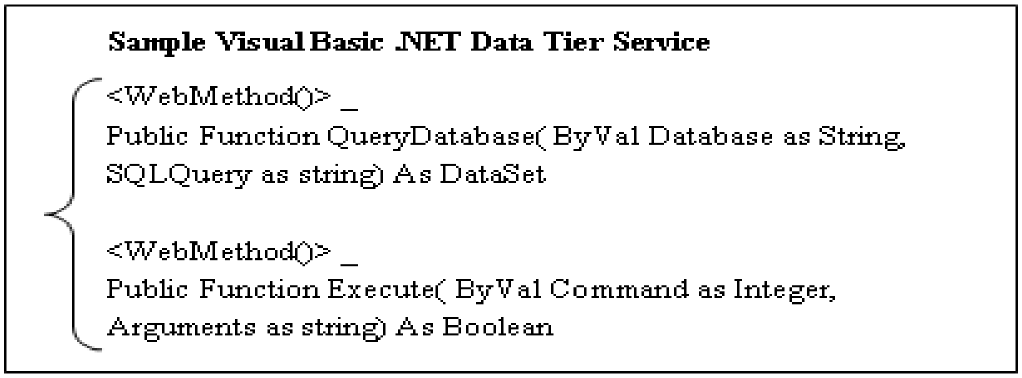


Figure 2.1 Service Architecture Model

- ✓ **Service description** is an entity that is defined as a collection of statements that describe the interface and the semantics of a service. .It contains details of the service's interface, its potential and expected behaviour. It includes a summary, such as, a job description, data types, operations, transport protocol information and address. It should also include classification, other metadata to obtain, facilitate discovery and utilisation (D. Austin, A. Barbir et al. 2004; S. Weerawarana, F. Curbera et al. 2006).
- ✓ **Service interface** is an entity that defines as an abstract border what a service discloses. It identifies the types of messages and the message exchange models that are included when interacting with the service, together with any conditions implied by those messages(D. Austin, A. Barbir et al. 2004; C. Matthew MacKenzie, Ken Laskey et al. 2006).

- ✓ **Service intermediary** is an entity that works in the background as a Web service whose main task is to alter messages in a value-added way. Particularly, the service intermediary is a service whose outgoing messages are equivalent to its incoming ones in some application-defined sense. Hence, the service intermediary is a specific type of service that typically performs as a type of filter on messages it handles (David Booth, Hugo Haas et al. 2004).
- ✓ **Service Role** is an entity that defines as a summary a set of tasks that are relevant to a user or an organisation offering a service. The service role is an intermediate notion between service and task. In addition, the service's roles are connected to particular phases in the messages that are exchanged with a service. The distinction can be formalised by noting that a service role is typically associated with a particular property of the message. Service roles identify the points of interest that a service owner has in the processing of messages(D. Austin, A. Barbir et al. 2004; David Booth, Hugo Haas et al. 2004).
- ✓ **Service Semantics** in a service is the conduct expected when acts interacting with the service. Semantics declare a contract between the provider entity and the requester entity. It liberates the intended real-world effect of invoking the service. Service semantics is the agreement between the provider and the requester entities regarding the outcomes and requirements referring to the use of a service. It illustrates the intended effects of using a service as identified by a service description. Service semantics may use a formal, machine-processed language. Knowing the type of data structure, however, is not enough to understand the goal and meaning behind its use. For instance, making a deposit/withdrawal into a bank account typically has the same type signature but with a different effect, the result of the operation is the semantics of the operation. It is good practice to be explicit about the intended effects of using a Web service even to the point of constructing a machine-readable description of the semantics of a service.

Machine processed semantic descriptions show the potential for sophisticated usage of Web services. For instance, by accessing such descriptions, a requester agent may autonomously choose which provider agent to use. Apart from the expected behaviour of a service, other semantic phases of the agreement will include any policy restrictions on the service, the relationship between the provider and the requester entities and what manageable features are associated with the service(D. Austin, A. Barbir et al. 2004; David Booth, Hugo Haas et al. 2004). Figure below shows an example of a tier service by MS VB.Net.


Sample Visual Basic .NET Data Tier Service

```
<WebMethod()> _
Public Function QueryDatabase( ByVal Database as String,
SQLQuery as string) As DataSet

<WebMethod()> _
Public Function Execute( ByVal Command as Integer,
Arguments as string) As Boolean
```

- **Service Task** is an entity that defines a form of business service with a functional framework that depends on a specific business process. In general, service task is not considered agnostic and thus has less re-use potential than other service models. Service task is a combination of actions that is connected to a preferred goal. Achieving the task involves completing the actions to achieve a particular aim. The service task has a service interface. It is also a concept that encapsulates some intended effect by invoking a service. Tasks are connected with goal states, which are portrayed by predicates that are satisfied on successful completion. The performance of a task is made observable by the exchange of messages between the requester and the provider agents.

Figure 2.2 shows exchange messages with several databases. Moreover, there may be other private actions related to a task, such as, a database update. The task could be pointed to by an initiation and completion message, which are public and the real database update, which is typically private.

In the case of service-oriented architecture, only the common phases of a task are important and are completely stated in terms of the messages that are exchanged. Service tasks are a useful unit in modelling the semantics of a service and indeed of a service's role. A given service may be formed from a number of tasks(David Booth, Hugo Haas et al. 2004; Hugo Haas and Brown 2004).

UOW: 123
Order: "A"

Figure 2.2 exchanging messages between different databases

- **Policy**

Information technology resources have become increasingly complicated and need to be controlled; therefore, administrators and managers must consider heterogeneous systems, different networking technologies and distributed applications. As the number of resources that are managed grows, the task of supervising these devices and applications depends on numerous systems and vendor specific issues (Xiaoyuan).

Policy is very common word that used to organise the rules. Policy in general, looks like a decision or a set of decisions. Decisions and policies are not themselves statements but they only a set of actions, although, as with decisions, we can infer what a person's or an organisation’s policy is from the statement he makes about it or if he makes no statement or we do not believe his statement from the way he acts. Similarly, we can claim that a statement or set of actions is misleading and does not faithfully reflect the "true" policy. Then, it is not like a decision(Verma 2000). The term policy usually implies some long-term purpose in a broad subject field. Nevertheless, we conceive that policy is not so much actively purposefully oriented but rather a cohesive set of responses to a problem that has arisen. In addition, policy is widely deployed in information systems and networking services viz. security, management and QoS. Policy is a proving solution for securing wide Web application systems(Hedi Hamdi, Adel Bouhoula et al. 2007).

Whenever a policy is defined there is a question of who and what is authorised to state the goals (permission) and carry out the necessary actions (obligation). So, policy is a collection of action that owing to decision and event of government intended to influence decisions, actions and other matters(Denis 2006). The meaning of policy is used to refer to the specific security rules of particular systems. It creates mandates, usually emanating from the highest authority of an organisation. .They are usually phase to allow for a certain degree of longevity. Policy must be formed and maintained as a living document and it needs to be able to grow along with any changes to an IT environment. In addition, it must clearly show and delineate its network context. Policy indicates the unified regulation of access to network resources and services based on administrative criteria.

Policy specifies the rule that allows a user to participate in the use of network resources and services. Policies control which users, applications or hosts should have policy architecture access to which resources and services and under what conditions.

In addition, policies, (Xiaoyuan) define the desired behaviour of resources. They are recognised as concepts that carry out complex management tasks by identifying the means that enable enforcement of this behaviour. Policies are created and organised at every level of a corporation forming a policy hierarchy, starting from corporate level through to small business units. At all levels, they state the desired behaviour of the underlying resources. At the corporate level, policies are primarily subjective and guided by institutions whereas policies for network and systems management are technology orientated. At this level, formalisation seems possible and necessary. Policies are important between interacting web services where they may use by a web service to notify users/clients about constrains under which it is operating, such as, the “system is down,” or the “system is under maintenance.”

Moreover, policies have wide statements that support and cover general security concerns. Hence, they are high-level descriptions that do not change regularly and are based on the organisation’s requirements(Sloman 1994). They are, therefore, a set of assertions e.g. obligations, rules and requirements based on a system that identifies properties of contract of web service communication. Policies are constrained the behaviour of system components. Policies are often request to apply to automated network administration tasks, such as, configuration, security, recovery, or Quality of Service (QoS) (Andrzej Uszok, Jeffrey M. Bradshaw et al. 2004).

- **Message**

Message is an entity that defines the purpose of a communication. It is something that presents information and can be this information itself. Therefore, its meaning is based on the context in which it is used. In addition, it is a brief report or statement made by a user or agent. Thus, it is an essential unit of data sent from one web service agent to another (David Booth, Hugo Haas et al. 2004).

- **User**

A user or consumer or an organisation is an entity that can request a service or may perform or run one. A service is an abstract resource that represents a means of performing duties and offers a coherent functionality for the provider and requester entities. To be used, however, a service must be realised by a concrete provider agent that acts on behalf of the person or organisation - the provider entity. Thus, a service is a thing that accomplishes a purpose. A service is like a worker that achieves a specific end-goal for a requester(Newcomer 2002). In order to successfully exchange messages the requester and provider entities ought to accept first both the semantics and the mechanics of the message exchange that will govern the interaction between the requester and provider agents. Next, the service description and semantics need to be understood by the requester and provider agents. Later, the requester and provider agents exchange messages, hence performing some task on behalf of the requester and provider entities(David Booth, Hugo Haas et al. 2004).

- **Requester and Provider**

The Web service acts as a requester and provider agent and so provides some functionality on behalf of its owner (e.g. a person or organisation). The provider entity is the user or organisation that presents a suitable agent to implement a particular service. A requester entity is a user or organisation that hopes to make use of a provider entity's Web service. It will apply a requester agent to exchange messages with the provider entity's provider agent(David Booth, Hugo Haas et al. 2004).

- **Agent**

An agent is a user or an entity that is authorised to take a certain action. The agent is either a set of software or a hardware component that is able to perform in a precise way in order to achieve tasks on behalf of its requestor. In addition, the agent is part of a programme that accomplishes some information processing tasks in the background (H. Janicke 2007).

- **Resources**

Resource is an entity that is defined as an accessible source of wealth. In general, any item, device or storage that can be used and shared or a new or reserve supply that can be accessed whenever it is needed. In addition, it can be used to specify many things. In general, resources are used to illustrate important materials. In addition, it can be used to show anything that is used to supply or resource something else. It refers to service capability, data, component or real world effect.

- **Service**

As a result, a service is a unit or an entity of logic solution that can perform tasks to which service-orientation is applied to a significant extent. The service occupies a position on the network so that it has a machine-readable description of the messages it receives and optionally returns. Therefore, the service is identified in terms of the message exchange model it supports. A schema for the data held in the message is used as the main part of the contract between the service requester and provider. Other items of metadata illustrate the network's address for the service, the operations it supports and its requirements for reliability, security and to carry out transactions (Newcomer 2002).

2.2.3 Behaviour

Behaviour refers to the actions that made by a systems and impact with its environment, which includes the other systems. It is the response of the system or organism to various inputs, whether internal or external, conscious or subconscious, covert. Behaviour is the way of responding to a system of the situation that been found. Behaviour is a set of values that alternates over time (Harel and Polit 1998).

Moreover, the term behaviour is classified in many senses such, the action or reaction of something as a machine under specified circumstances, behavioral attributes the way a person behaves toward another person and psychology the aggregate of the responses or movements made by an organism in any situation.

The service behaviour is set of collection of data that includes information of services which should be used for control the system. It is a sequence of state of the system that may impact its environment or system.

Behaviour service is a description of sequence of state of the service that specifies dynamic aspects of entire system. It specifies the states and modes of the system that could impact with its environment.

2.3 Quality of Service (QoS)

With the fast advent of Internet, its usage and complexity, offering a Quality of Service (QoS) which becomes increasingly important for network design, traffic, troubleshooting and service-level-agreement (SLA) verification. Further, the rapid development of the Internet makes QoS monitoring a challenging task. QoS is the most important issue when determining the efficiency of a Web service. As businesses start to create new functionality in the form of composite Web services, QoS becomes increasingly more important(Xiaoyuan 2007). QoS is one of the most elusive and confusing fields to date pertaining networking. It (Paul Ferguson and Huston 1998) allows user to offer a better service flow. Moreover, it describes the assurance of sufficiently low delays and packet loss from different types of applications or traffic. The term of QoS relates to resource reservation control mechanisms regardless of the achieved service quality. QoS is the capability to present different priorities to different applications its users to be ensured of a smooth flow of data (Paul Ferguson and Huston 1998). One definition of QoS refers to a diversity of methods that prioritise one type of traffic or a programme that runs across a network connection instead of being only based on good effort connectivity. QoS also ascribes to the capability of a network to give a better service to selected network traffic over different technologies. The main aim of QoS is to provide priority containing dedicated bandwidths, controlled jitter and latency and mitigate loss characteristics(Andrew, Geoff et al. 1994; Corp 2003; Boxman 2005).

QoS has been the major subject of research in packet networks in recent years (Foster and 1998; Paul Ferguson and Huston 1998). Concepts of QoS were primarily used in networking and multi-media applications. There has been a surge in adapting this concept to Web services in recent years(Steve 2003).

QoS caters for estimable qualities, such as, latency and throughput, things that directly influence user experience. Generally, network traffic and packets are handled in a “best effort” manner. QoS is sometimes referred to as traffic control or traffic shaping (Boxman 2005). It is a wide term utilised to express the overall experience that a user or application could receive over a network. An advantage of QoS is that it includes a wide range of technologies, architecture and protocols and a network should have end-to-end QoS so that network component requests are guaranteed consistent treatment with respect to traffic flows across the network.

In addition, QoS is about choosing the correct set of resources, applications or users that allow high priority access to network resources(Raju Rajan, Dinesh Verma et al. 1999).

2.3.1 Quality of Service Architecture (QoS-A)

QoS-A (Andrew, Geoff et al. 1994) presents a framework for indicating and implementing the required performance properties of multi-media applications over high performance on the network. As shown in figure 2.3 QoS-A is a set of layer architectures of services and mechanisms for QoS management to control the continuous media flows in multi-service networks.

The essential architectural concept that is used is the notion of flow. A flow distinguishes the production, transmission and eventual consumption of a single media stream as an integrated activity governed by a single statement of QoS. Flows are always simplex but can be either uncast or multicast. They may carry a range of data types that include continuous media and control data, such as, messages or RPC packets. The realisation of the concept of flow requires active QoS management and tight integration between device management thread scheduling, communications protocol and network components of the end-to-end data path.

The figure 2.3 below depicts different levels at which regulation may be expressed and exercised(Raju Rajan, Dinesh Verma et al. 1999). They are a convenient mechanism to control and change the system's behaviour.

However, policies have to be used to specify how a management system could or could not authorise policies; for example, a policy can be used that limits attempts to login to no more than three attempts and that should be authorised by the system. Policies illustrate a variety of written sources that express security practices within an organisation. Policies are declarations that reflect an organisation's approach toward security and how it affects (Sloman 1994).

In functional terms, in figure 2.3 the QoS-A is shown broadly divided into various layers, roles and planes. The upper layer contains a distributed applications platform augmented by services that present multi-media communications and a QoS configuration in an object-based environment. In the layer below, however, the platform level is an orchestration one, which offers multi-media synchronisation services across multiple related application flows with jitter correction. Supporting this is a transport layer, which includes a range of QoS configurable protocols. For instance, separate protocols are given for continuous media and constrained latency messages protocols.

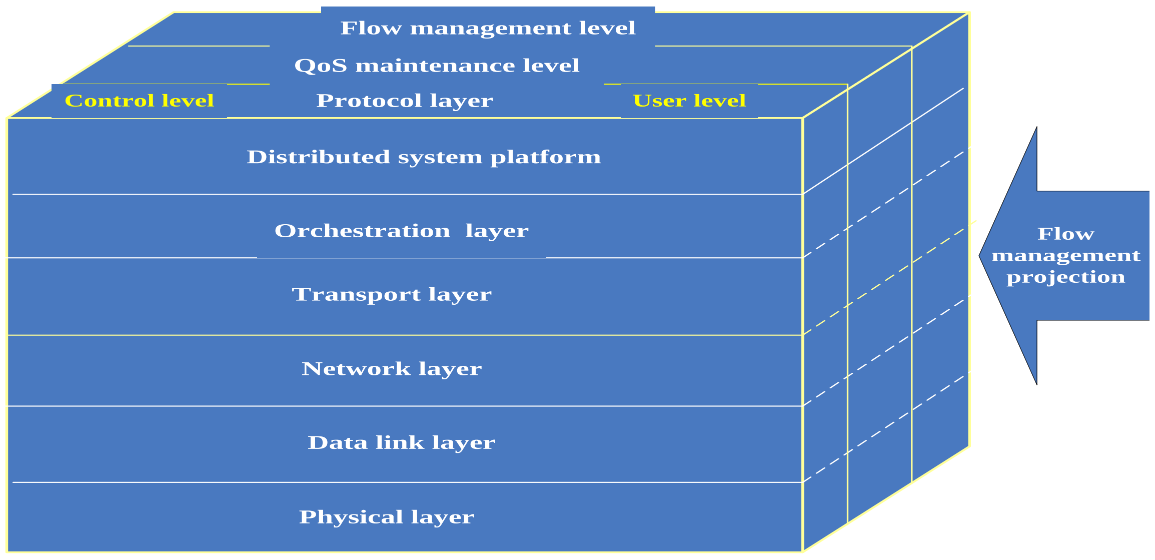


Figure 2.3. QoS Architecture

The vertical planes of QoS-A have three levels as follows(Andrew, Geoff et al. 1994; Mario 2007):

- The protocol level: contains a user and a control level. The QoS-A, uses separate protocol profiles to control the flow of data components because of the primary different QoS requirements for control and data. In general, control needs a low latency full duplex assured service whereas multi-media data generally requires a range of non-assured, high throughput simplex services.

- QoS maintenance level: contains many layer particular QoS managers. They are each responsible for fine-grained observing and maintenance of their associated protocol entities. Based on flow observing information and a user supplied service contract, QoS managers support the level of QoS in the managed flow by means of fine-grained resource tuning strategies.

- Flow management level: is responsible to establish a flow (including flow admission control, resource reservation and QoS based routing, QoS re-negotiation, QoS mapping (which translates QoS representations between layers) and QoS adaptation (which implements coarse-grained QoS maintenance control). As shown in the QoS-A, flow management projection demonstrates the relationship between the three levels, which work together to supervise and examine end-to-end Qos.

2.3.2 Basic QoS management

QoS management is a set of assistant and evaluations to help QoS policies and goals. Figure 2.4 shows a basic arrangement for QoS management. It presents three essential components for QoS implementation(Systems CISCO 2001):

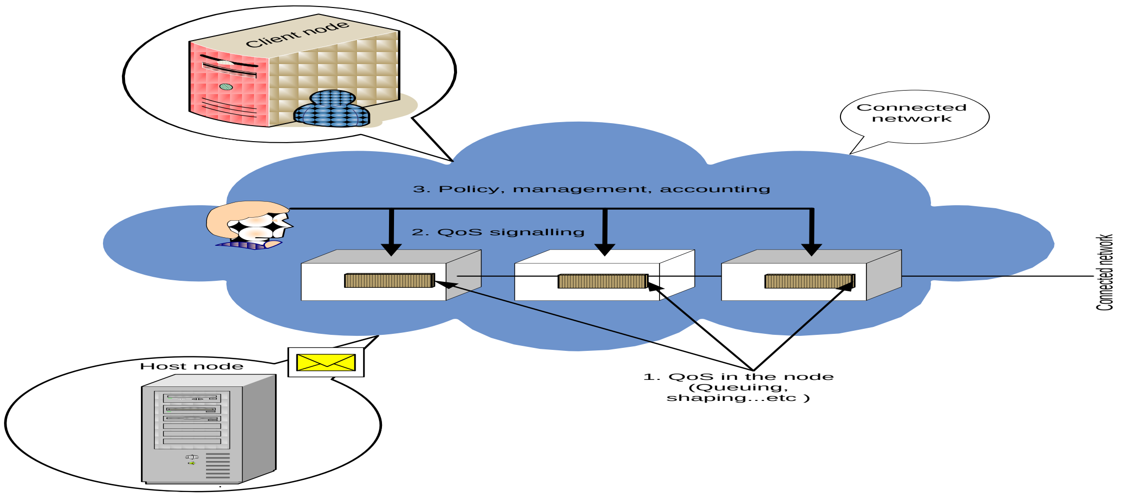

Client node
Connected network
3. Policy, management, accounting
2. QoS signalling
Connected network
Host node
1. QoS in the node (Queuing, shaping...etc)

Figure 2.4 Basic QoS management

- QoS classification and marking techniques are used to co-ordinate QoS from source to destination between network factors.

Explanation

QoS identification and marking is achieved through classification and reservation. Classification offers a desirable service for a given type of traffic but firstly it should identify. Secondly, the packet may be marked or unmarked. These two tasks precede classification. When the packet is identified but not marked classification is on a per-hop basis. When packets are indicated or marked for network-wide use IP precedence bits can be set.

- QoS within a single network aspect viz. queuing, scheduling and shaping

Explanation

QoS within a single network aspect, for example, congestion management, queue management, link efficiency and shaping/policing tools.

Congestion Management

Due to the huge volume of voice/video/data traffic, the amount of traffic sometimes surpasses the speed of a link. In this case, the router will buffer traffic in a single queue and enable the first packet to be selected or it will assign packets into different queues and service them more often.

- ✓ Queue Management

As queues are of finite size, they could fill and overflow. Then, additional packets cannot be taken into the queue and will be dropped. The issue with tail drops is that the router cannot avoid this occurring.

- ✓ Link Efficiency

On numerous occasions, low-speed links present an issue for smaller packets. This may occur for a number of different reasons, for example, due to the cable, router or modem issues. An ADSL or broadband connection gives more efficiency in this case.

- ✓ Traffic Shaping and Policing

Shaping is used to generate a traffic flow that limits the full bandwidth potential of the flow(s). Policing is similar to shaping but differs in one very important way. Traffic that surpasses the configured rate is not buffered and is normally discarded.

- ▪ QoS policy, management and accounting functions that organise and manage end-to-end traffic across a network.

Explanation

Service levels apply to actual end-to-end QoS capabilities. Services will differ in terms of their level of QoS strictness, which expresses how much the service can jump due to specific bandwidth, delay, jitter and loss characteristics.

In addition, QoS (M. Papazoglou, P. Traverso et al. 2006) is surrounded by the important features of functional and non-functional service quality attributes, such as, performance metrics, accessibility, security, integrity, reliability, scalability and availability. Having and delivering QoS on the Internet is a meaningful and significant challenge regarding its dynamic and unpredictable nature. Software applications with very different characteristics and requirements compete for all types of network resources. The creation of Internet QoS standards is required for changes in traffic models, securing mission critical business transactions, the effects of infrastructure failures, low performance of Web protocols and reliability issues over the Web. Frequently, unresolved QoS issues make critical transactional applications suffer from unsatisfactory levels of performance degradation. Usually, QoS is estimated by the grade to which applications, systems, networks and all other essentials of the IT infrastructure support availability of services at a needed point of performance under all access and load conditions. While traditional QoS metrics apply, the attributes of Web service environments consider two things. High availability of applications and raised complexity in terms of accessing and organising services and hence impose specific and intense demands on organisations, which QoS must address.

2.3.3 Web service and QoS requirements

Often, the Web services (Mani and Nagarajan 2002) that are deployed cannot offer guarantees for QoS. The recent growth of Web services and QoS has become a major issue in characterising the success of service providers. QoS decides both of the service usability and utility influence the popularity of the service.

There is a vision of dynamic e-business requests in a seamless integration of business processes, applications and Web services on the Internet. The main challenge here is delivering QoS over the Internet. Delivering QoS has become a crucial aspect due to its dynamic and changeable nature.

All major Web service players are approving Web standards, such as, SOAP, UDDI and WSDL. These include those Web services that cover the financial business services, high-tech and media. Web standards that serve these sectors are being improved. These Web services will need to be launch and adhere to standards, therefore, QoS will be become a significant selling and differentiating key between these services. QoS covers a whole range of techniques that join the needs of service requestors to those of service providers based on an availability of the network resources.

2.4 Service-Oriented Computing-SOC

The landscape of today's business technology is changing. Traditional integrated enterprises with centralised control are able to loosely-coupled network applications that are controlled, owned and managed by diverse business partners. A vision of SOC technology is ongoing to achieve its rhetoric. An SOC (Tsai and 2006) pattern is a set of notions, principles, techniques and methods that position computing in a Service-Oriented Architecture (SOA) where software applications are constructed based on independent component services with standard interfaces. SOC (Stefan, Rania et al. 2004) is a distributed computing paradigm that deals with the distributed, loosely coupled and heterogeneous nature of this trend in a first-class manner.

The major utopia of SOC and SOA is to fragment explicitly software engineering from programming in order to highlight on software engineering, and to de-emphasise the latter. SOC splits software development into three independent parties(Edgardo, Marco et al. 2006):

- Application builders (by software engineers)
- Service providers (by programmers)
- Service brokers (Co-operative effort from standard organisations, computer industry and government).

Service providers use a computer programming language, such as, C++, C# or Java to write programme (code) components. All components will be wrapped with open standard interfaces, call Web services if they are available over the internet. This means that application builders can use the services without further communication with their service providers. In addition, the same services can be used by many other applications.

Service brokers: permit services to be registered and published for the public to access and use. Help and facilitate application builders are able to find services they need.

Application builders: instead of constructing software from scratch using old or basic programming language constructs, such as, Pascal or Borland C, application builders enable end users to specify the application logic in a high-level specification language, such as, object-oriented language by using standard services as components. Application builders are software engineers or software planners who have a good understand of software architecture and the application domain.

SOC allows the creation of customised software in a dynamic environment depended on re-usable services with well-defined interfaces that are available on the Internet(Gorton and Reiff-Marganiec 2007). It is an emerging model for distributed computing and service processing. It uses services as vital elements to enable the building of agile networks of collaborating business solutions distributed within and across organisational boundaries (CÖMERT 2004; Huhns and Singh Feb 2005). In addition, it is a recent idea for a computing pattern that employs services as important factors for increasing networked applications (Nirmal, Ravi et al. 2004). In short, it is a modern ideal of a computing service that uses basic constructs to support rapid low-cost growth and easy composition of distributed applications in many heterogeneous environments. The expectation of SOC is a world of co-operating services where application components are assembled with little effort into a network of services that can be loosely coupled to create flexible dynamic business processes and quick applications that may span organisations and computing platforms(M. Papazoglou1, P. Traverso et al. 2006). SOC (Papazoglou 2003) is a computing hypothesis that supports services as primary elements for enhancing applications or solutions to make the service model. SOC is based on SOA, which is a way of re-arranging software applications and infrastructure into a set of interacting services.

However, an essential or basic SOA could not refer overarching concerns e.g. management, service orchestration, service transaction management and coordination, security as well as other concerns that apply to all components in a service's architecture.

Developers and designers think that SOC has become a valuable area of research and improvement. Business management continues to favour SOC-type architecture, which could support and improve information technology (IT) in order to enhance and develop their systems and architecture. SOC allows applications to integrate faster and easier. This integration occurs at an advanced stage in the protocol stack, based on messages centred more on service semantics and less on network protocol semantics, which enable a loose integration of business functions. The integration of applications can be increasingly based upon available platforms, languages and by adopting existing legacy applications(CÖMERT 2004; Huhns and Singh Feb 2005).

The advance of Web services technologies offers to have far-reaching effects on the Internet and enterprise networks. Having Web services technologies (K. Gottschalk, S. Graham et al. 2002) means language and environment-neutral programming models that express application integration inside and outside enterprises and organisations.

WSs (K. Gottschalk, S. Graham et al. 2002) can easily applied as a wrapping technology around existing applications and information technology (IT) assets, therefore, new optimal solutions can be increasingly deployed and re-created to address new opportunities. As the adoption of Web services(K. Gottschalk, S. Graham et al. 2002) accelerates, their number of will increase, which will foster the development of more dynamic models of just-in-time applications and business integration over the Internet.

WSs based on the idea of SOC are a most promising technology. They support e-business on the Internet via standard protocols and interfaces. Web service technology enables the composition of less dynamically changing forms of the system. WSs in various forms are widely becoming a significant emerging technology that is still undergoing change and has yet to mature(Jeffrey, Marijn et al. 2004). The crucial advantage of Web services is their machine-to-machine interaction (Alter 2005). WSs allow applications to be integrated faster, easier and cheaper. The ongoing research of SOC is a world of supporting services loosely coupled to flexibly create dynamic business processes and agile applications that may extend across organisations and computing platforms and that can quickly and autonomously adapt to changes of context and requirements. SOC exploits services as essential elements for developing applications and solutions (Bussler 2002; Petrone, Ardissono et al. 2003; Cavanaugh 2006).

The ability to exchange information between internal enterprise business units, partners and customers is essential for success. Several research efforts continue to focus on a variety of phases of SOC, particularly web service technology, composition, specification modelling, discovery and verification(Jianchun and Subbarao 2005). WSs are the most promising technology for easier system integration by giving standard protocols, such as, XML messages for exchanging data and a standard interface declaration language viz. the Web Service Description Language(WSDL) (Nirmal, Ravi et al. 2004; Avik and Amit 2006) (Petrone, Ardissono et al. 2003).

However, for e-businesses, a Web services application (K. Gottschalk, S. Graham et al. 2002) needs to meet the strict demands of trading, therefore, an enterprise-class infrastructure must be supplied that includes three aspects, namely, security, management and quality-of-service management. As well as, e-businesses valuable Web services applications have also been developed and deployed but widespread commercial exploitation of Web services across the public Internet awaits development and acceptance of higher-level standards in such areas as security, reliable messaging, transaction support and workflow(K. Gottschalk, S. Graham et al. 2002).

In recent years, Web services have rapidly expanded to become more popular with application developers. WSs can be used, run and developed for any platform environment regardless of the operating system. They can exchange and communicate with other Web services using common protocols. They make communication much easier for trading partners to connect by modern electronic means. WSs overcome the obstacle of various platforms and operating systems and create new business opportunities and a greater degree of business performance flexibility (Goethals 2002; Newcomer 2002; Cavanaugh 2006; Jagadeesh, Nandigam et al. 2006).

2.4.1 Meaning of Service-Oriented Computing

Munindar and Huhns, M. (M. Papazoglou and Georgakopoulos 2003; Stephen, James et al. 2005; Huhns and Singh Feb 2005) define SOC as,

> *"SOC is a process of discovering and composing the suitable services to satisfy a specification. It is a computing paradigm for distributed computing that is changing the way of software applications which are designed, delivered, consumed and architected. It involves extended, loosely coupled activities among two or more autonomous business partners."*

The purpose for a SOC is a worldwide mesh of collaborating services, which are published and available via its capabilities. Having and adopting SOC is essential to deliver business agility and IT flexibility as promised by the Web Services. These benefits are delivered not by just viewing their service architecture from a technology perspective and the adoption of Web Service protocols(David Sprott and Wilkes 2004).

2.4.2 Advantages of Service-Oriented Computing

SOC has a collection of methods, principles and concepts that show and represent computing in SOA where software applications are constructed based upon independent elements and component services with standard interfaces(Ravi and Pallavi 2006). SOC has some advantages that offer a number of research challenges that need to be addressed. These include among other things, integration, monitoring, composition, the discovery of services and their quality, development, evolution and security. These issues are attracting the interest of researchers in different communities, such as, databases, software engineering, artificial intelligence and distributed systems. The advantages of SOC are articulated as (Nirmal, Ravi et al. 2004; M.Huhns and 2005; Nabor, Jos et al. 2005; Tsai and 2006; Huhns and Singh Feb 2005).

1. SOC has a significant economic advantage that facilitates application developers rapidly and dynamically to grow application portfolios more than ever before by creating compound application solutions that use internally existing organisational software assets, which they appropriately join with external components possibly residing in remote networks.

2. SOC allows novel flexible business applications of open source systems that would not be possible otherwise.

3. SOC enables the customisation of new applications by offering a Web service interface that removes messaging problems by giving a semantic basis to customise the functioning of the application.

4. SOC in open source systems develops the production of programming and administering applications, viz. applications are notoriously complex.

5. SOC offers the efficient usage of grid resources and facilitates utility computing, especially where huge services can be used to achieve fault tolerance. SOC presents a semantically rich and flexible computational model, for which it is easier to produce software.

2.4.3 What are Web Services?

A Web service (K. Gottschalk, S. Graham et al. 2002), as the name pronounced, is an interface that depicts a collection of operations and processes that are network-accessible and accessible through standardised XML messaging. A Web service performs a specific task or a set of tasks. WSs (Newcomer 2002) offer and present a layer of abstraction over existing software systems, such as, CORBA, .NET servers, messaging, and packaged applications. WSs, for example, XML, WSDL, SOAP and UDDI unlike existing distributed computing systems, are adapted and accomplished on the Web. HTTP is the default network protocol. Most existing distributed computing technologies contain this communications protocol as part of their domain. With Web services, the communications protocol is already in place, even for the far-flung. Furthermore, WSs are capable of bridging any operating system, hardware platform or programming language. It is possible, therefore, easy to develop new software applications, as everything is Web service enabled. The advent of web services means that existing business patterns, discussion groups, interactive forums and publishing models will adapt and new ones will emerge to take advantage of this new capability. WSs are rapidly emerging as a popular standard for sharing data and functionality between loosely coupled and heterogeneous systems. Currently, most organisations employ a diversity of disparate applications that exchange and store data by different approaches(Utkarsh, Kamesh et al. 2006). WSs support the basis for e-business processes that are distributed over the internet and exist via standard protocols and interfaces (Jianchun and Subbarao 2005). WSs are SOC implemented and provide a simple mechanism to connect applications regardless of the type of device that is used or its technology. WSs have evolved as a practical, cost-effective solution for uniting information distributed between critical applications over language barriers, operating systems and platforms that were impassable(Stephen, James et al. 2005).

Moreover, WSs are a collection of promising and established communication protocols that includes of XML, Simple Object Application Protocol (SOAP), Universal Description Discovery and Integration (UDDI) and Web Services Description Language (WSDL) over Hypertext Transfer Protocol (HTTP) (Jeffrey, Marijn et al. 2004; Cavanaugh 2006). WSs allow applications to combine faster, simply and inexpensively. They are used and expressed as a WSDL that is XML-based language (Stephen, James et al. 2005). These services will specify a contract between the client and the operations that the user can expect. The services could be published, used and discovered using UDDI, whilst SOAP allows vendor-neutral communication between applications over HTTP (Coyle 2002; Newcomer 2002; Katia Sycara, Paolucci et al. 2005; Jagadeesh, Nandigam et al. 2006; Ravi and Pallavi 2006).

2.4.3.1 Meaning of Web Services

WSs are achieving momentum, thus they have become widely established for use in different activities on the Web. They promise to be the next wave of innovation in the Web revolution (Shalom, Serge et al. 2001). WSs, also called service computing, are uncomplicated and self-contained applications, which achieve functions from easy requests to difficult business processes. They happen through a machine-to-machine communication without a user interface to call the services. The Meaning of Web services is given by(Bussler 2002) (Goethals 2002) as follow:

"Web services are a new breed of Web application. They are self-contained, self-describing and modular applications that can be published, located and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes. ... Once a Web service is deployed other applications (and other Web services) can discover and invoke the deployed service."

The WWW Consortium (D. Austin, A. Barbir et al. 2004) has defined Web services as a software application identified by a Uniform Resource Identifier (URI), whose interfaces and binding are capable of being defined, described and discovered by XML artefacts and supports direct interactions with other software applications using XML based messages via Internet-based protocols. Currently, Web Service based computing is an essential driver for the software management business. WSs are software components that communicate using pervasive, standards-based Web technologies including HTTP and XML-based messaging(Massimo, Mourad et al. 2006). Next, they are efforts to extend the Web from an infrastructure that provides services to humans to one that provides services to software looking to connect with other software. This innovation can be developed on any platform and within any developmental environment and communicate with other Web services by using the above common protocols (Sleeper, and et al. 2001; Bussler 2002; Tian, et al. 2003).

2.4.3.2 Web Services Architecture (WSA)

Web services architecture (Ethan 2002; Frank 2002; Stefan, Rania et al. 2004; Stephen, James et al. 2005) aims to provide and offer a standards-based platform for SOC. It defines a set of specifications that support an open XML based platform for the description, discovery and inter-operability of distributed heterogeneous applications as services. It is based on the interactions between three roles as shown in figure 2.5 (Bussler 2002; Edgardo, Marco et al. 2006; Ravi and Pallavi 2006). These are the Service Provider, the Service Registry and the Service Requestor. The interactions involve publishing, finding and binding operations. Together, these roles and their operations act upon Web services artefacts, the Web service software module and its description. It (Frank 2002) places the above into a relationship among various components and technologies. It forms a Web Services “stack,” which is capable of complete functional implementation. The basic architecture includes Web services technologies that:

- Exchange messages
- Describe Web services

- Publish and discover Web services descriptions

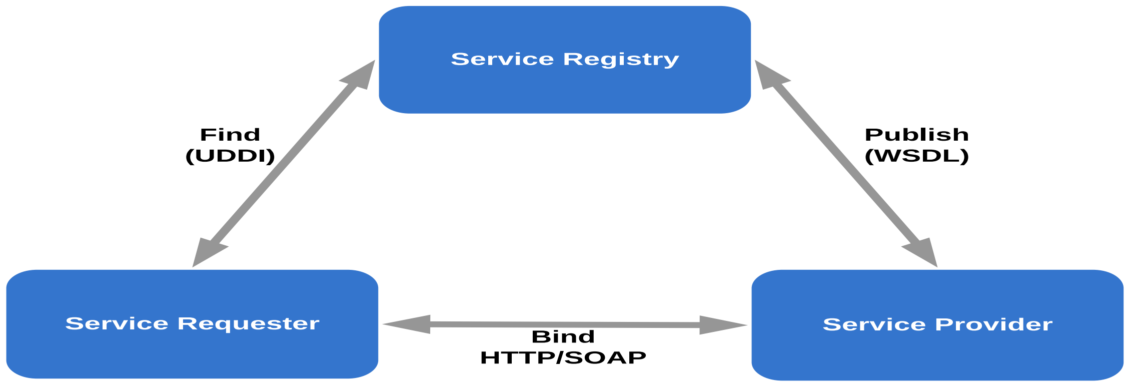


Figure 2.5 Web Service Architecture

2.4.3.3 Scenario of Web services

As illustrated in figure below 2.6, the scenario involved with Web services is that service providers deploy and publish services to the service registry. The service requester using a service broker explores the services available and subsequently negotiates with service providers to bind them. The service requesters can be a human user, a client, a device, an application or any other web service. A service broker offers registries for exposing web services. A service provider and service requestor’s roles are logical constructs and a Web service can demonstrate characteristics of both (Myerson 2002; Nirmal, Ravi et al. 2004; Jianchun and Subbarao 2005; Edgardo, Marco et al. 2006; Tsai and 2006).

- **Service Provider**:

A service provider is the owner of the service and is responsible for publishing a description of its service to a service registry. It also hosts the service and controls access to it.

- **Service Registry (broker):**

A service registry is a central store that makes possible service discovery by requestors. This component provides a searchable repository of service descriptions where providers publish their services and requesters find services and obtain binding data for these services.

- **Service Requestor:**

A service requestor is a software component in search of a service to invoke across the Web. It finds a suitable service by discovering the set of available services that meets some pre-defined criteria.

- **Publish**: Is how the providers of Web services registers themselves
- **Find**: Is how an application finds a suitable Web service
- **Bind:** Is how an application connects to and interacts with a suitable Web service once it has been found (Vasudevan 2001; Frank 2002)**.**

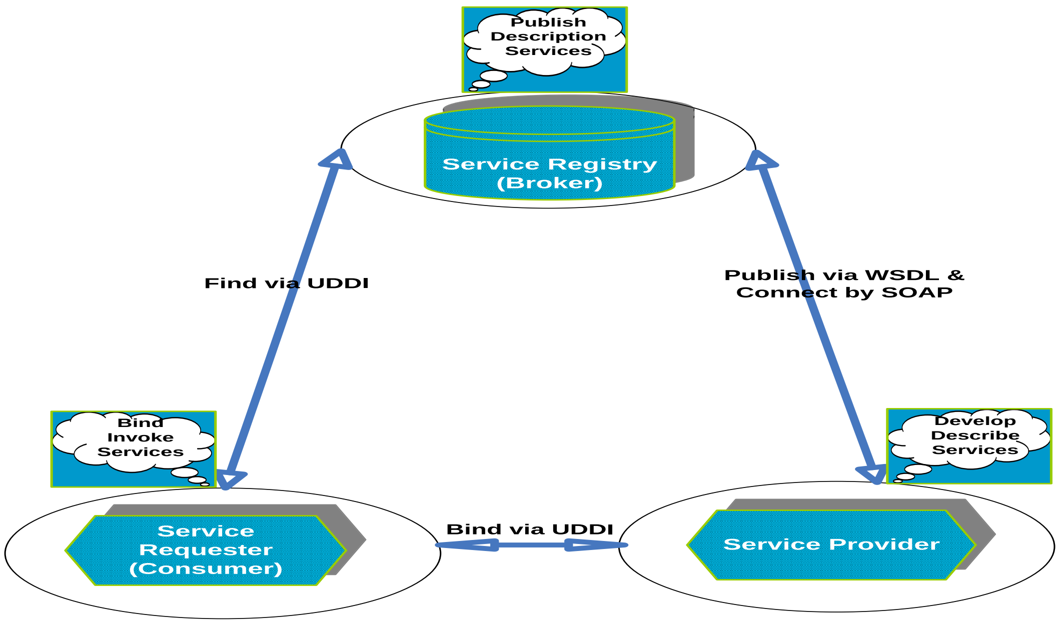

Publish Description Services
Service Registry (Broker)
Find via UDDI
Publish via WSDL & Connect by SOAP
Bind Invoke Services
Develop Describe Services
Service Requester (Consumer)
Bind via UDDI
Service Provider

Figure 2.6 Web service Scenario

Web services environments are pieces of business logic accessible via the Internet using open system standards. WSs are a group of emerging and established communication standard protocols e.g. XML over HTTP (CÖMERT 2004; Nirmal, Ravi et al. 2004; Jianchun and Subbarao 2005; Huhns and Singh Feb 2005). WSs have a major advantage that they inter-operate between several software applications running on a variety of platforms (Avik and Amit 2006) (Jianchun and Subbarao 2005) (CÖMERT 2004). Recently, WSs permitted a collection of solutions/applications to be integrated faster, easier and cheaper (Petrone, Ardissono et al. 2003). Further, WSs provide a standard means of inter-operating between different software applications that run on a variety of platforms or frameworks. WSs are independent software systems that can be promoted, located and accessed via messages encoded under an XML based standard, such as, SOAP, WSDL and UDDI (Petrone, Ardissono et al. 2003) and transmitted using Internet protocols (Cavanaugh 2006). Moreover, they encapsulate application functionality and information resources and make them available via programmatic interfaces, as opposed to the interfaces provided by customary web applications, which designed for manual interactions. WSs are designed to be discovered and used by other applications across the web. WSs need, therefore, to be portrayed and understood in terms of functional capabilities and QoS properties (Goethals 2002; Newcomer 2002; Jian 2003; Jagadeesh, Nandigam et al. 2006).

In addition, the appearance of WSs for ordering a purchase, finance, accounting, human resources and manufacturing has created unprecedented opportunities for organisations to develop agile and multi-collaborations with other organisations(Goethals 2002; Newcomer 2002). WSs are a collection of protocols, processes and standards that are used to exchange data between different applications or systems. WSs are easy to implement with existing technologies. Again, WSs are parts of software components that join using pervasive standards-based web technologies including HTTP and XML-based messaging (Papazoglou 2003). The effect of this is to expand the web from an infrastructure that present services to humans to one that presents services to software looking to communicate with other software. WSs can become enhanced to run on any platform and by any development. In addition, it can connect with other Web services using common protocols (M. Papazoglou and Georgakopoulos 2003),(Massimo, Mourad et al. 2006) (Nabor, Jos et al. 2005). In addition, different pieces of software are enabled to be executed in different languages and run on a variety of operating systems in a cheaply and easily. Applications that run in different parts of an organisation and in different ones come to exchange data easily and inexpensively. WSs are applications or solutions that run anywhere on any technology or device that has a web service. They are an automated innovation and present a mechanism for discovering service providers, which can be automated (Nirmal, Ravi et al. 2004; Jianchun and Subbarao 2005). They are defined as software with explicit interfaces and they remain ready to interact with other ones. To describe Web services, several standards have been widely adopted. These standards pertain to describing, publishing, enacting and composing services. They include WSDL (Richards 2006), UDDI (Panagiotis 2006) and so on. Besides that, with the popularity and proliferation of communications, services are now provided for devices, such as, a commodity. For developers, researchers, designers and vendors, web services offer a means to present integration points with their systems by synchronous and asynchronous message exchanges. Generally, web services technology is available with implementations in several widely used computer programming languages, for instance, Java, C++, C#, Perl, and Python (Newcomer 2002).

2.4.3.4 Future of Web services

WSs are expected as most technological change that will develop and revolutionise business. Most technology, business companies and visionaries have started their rhetoric and challenge with Web services. An entirely (Newcomer 2002; Vincent 2002) new era is emerging where anyone can publish their services using standard Internet protocols and consumers around the world can easily combine these services in any fashion to provide higher order services. These service network chains will solve every business problem in the world and in the process generate revenues to everyone involved in the chain. Despite the fact that the technology for implementing web services is available together with its capability the question arises, to what extent is it being used. When the whole world is moving in one direction, the way to use it is initially via the Web. This proves difficult when the change is in the field of technology. Every change is unique and can alter the whole of history. There will be some data points and issues that could be gathered by research and experience, which can be used to extrapolate future trends in technology. The future of web services can be predicted by comparing it with the real world of service interactions. As evident from real world services, one can expect only a certain type of service relationship to form sustainable business models.

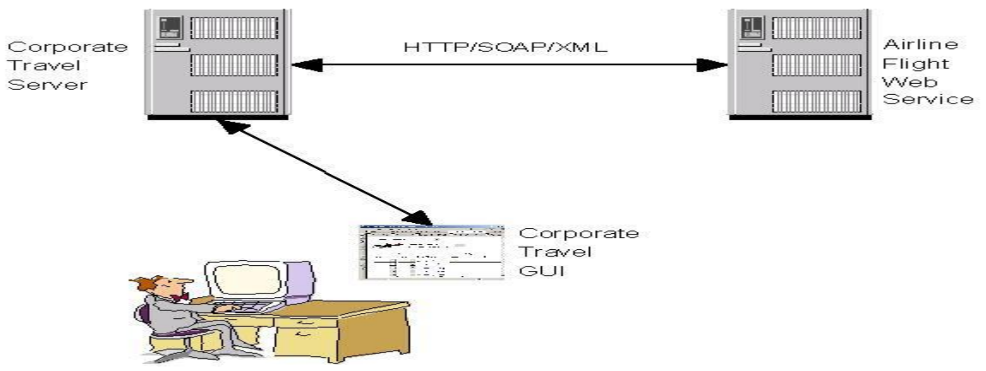


Figure 2.7 The next generation of the Web

Yet, the next generation (Newcomer 2002) will witness improved integration of the Service Web based on software-oriented interactions. Web services have the potential to put the vast global network of the Web established for human interaction, to a completely new purpose. Software-oriented interactions will automatically make operations that previously required manual intervention, such as, searching for and buying goods and services at the best price, co-ordinating travel tickets and restaurant tables for a given date, streamlining business procurement, invoicing and shipping operations as illustrated in the above figure 2.7, the next generation of the Web will use software-oriented services to inter-operate directly with applications built using any combination of objects, programmes and databases. However, Web services are not only about interfaces to objects, programmes, middleware and databases for access over the Internet. Combining a series of Web services into a larger interaction it will provide the means to perform new types of communications.

Most the Web's (Davies, Fensel et al. 2004) success can be ascribed to its simplicity. It directly offers a means by which static information could be published and interconnected on universal basis. The Web Services proposal effectively adds computational objects to the static information of yesterday's Web and offers distributed services capability over a network. They have the potential to produce new paradigms for both the delivery of software capabilities and the models by which networked enterprises will trade. In the meantime, Web Services technology, useful though it is, will be enhanced over the time by the harnessing of Semantic Web technology to deliver a step change in capability. In addition, they provide an easy way to make existing components available to applications via the Internet.

Currently Web services are essentially described using semi-structured natural language mechanisms, which mean that considerable human intervention is required to discover and combine Web Services into end applications. The Semantic Web will facilitate the accessing of Web resources by semantic content rather than just by keywords. In spite of technology (Vincent 2002) SOAP, XML and WSDL will continue to penetrate as technologies for joining software applications in a loosely coupled manner. The use of these technologies for building successful business models is altogether different. Successful business models for web services must satisfy many criteria. In the end, only the existing physical services make sense even as web services most of the time. There will be many constraints on pure software based web services not founded on real world services. Consequently, there needs to be further technological research accompanied by some unified industrial agreements on identification, authentication and development in the form of a security infrastructure before dreaming to write a piece of code and the whole world accessing it and fetching revenues for the creator.

2.5 Summary

In this chapter, we introduced, described and discussed Service Oriented Computing that provides separate tiers for composing, for co-ordinating and for managing services in an open marketplace by employing grid services.

Web services primary focus on document-level integration and virtualisation. The Web services approach abstracts the details of underlying platforms and operating systems, which reduces system coupling and much of the tight co-ordination and synchronisation required by traditional integration methods. It is more important than ever to build security into the co-operating systems and services early and throughout the software’s development life cycle. This is due to the complexity of co-ordinating (and achieving compatibility) of security policies and the requirements derived from these policies (as reflected, for example, in a common message schema) across all of the participants in a web services application.

This chapter has elaborated architecture for Service Oriented that provides optimal services for composing, for co-ordinating and for managing services in an open environment by utilising Web services. In addition, this chapter has illustrated the key point about SOC and its associated framework, such as, service discovery, service description and SOA. SOC involves extended, loosely coupled activities among two or more independent business partners, e.g. activities can be thought of as business processes that engage several services in a manner that brings about the desired business outcome. Adopting and having SOC has the potential to bring about a decrease programming difficulty expenses, lower costs, quicker time-to-market, new revenue streams and improved operational efficiency.

Further, this chapter has provided a survey of Web services, which are fast becoming an important technology in the evolution of distributed computing over the Web. These technologies are rapidly changing and a long list of additional features and functionality is required to complete the vision. Web services allow applications to become integrated faster, more easily and more cheaply than ever before and they can influence the data independence of XML to solve problems. The essential Web services standards SOAP, WSDL and UDDI are directly valuable for many applications, such as, publishing interfaces for automated business processes, bridging disparate software domains and connecting wireless customers for Web purposes. Nonetheless, there is vast interest in the Web about policy-based techniques as a means of implementing adaptive QoS management, caching, persistence and security to support modern software applications and omnipresent computing. QoS is a vital requirement of business-to-business transactions and thus a necessary part of Web services. The various QoS properties, such as, availability, accessibility, security, integrity, performance, need to be addressed together with the considered implementation of Web service applications. The properties become even more complicated when adding the need for transactional features to Web services. Some of the limitations of protocols, such as, *HTTP and SOAP* could delay QoS implementation but there are a number of ways to offer proactive QoS in Web services. QoS presents differentiated services, such as, higher-priority to flow services that offer an assured service level. Both of these are contrasted by best-effort services, which are provided by what is generally considered a lack of QoS. FIFO shows best-effort service where flows are not differentiated and are serviced on a first-come, first-served basis. We conclude, when we look at the various Web service QoS requirements, bottlenecks that affect the performance of Web services, approaches to provide service quality, transactional services and a simple method of measuring response time of your Web services using the service proxy.

CHAPTER 3

Critical Review of Security Policy

Objectives

- *To present a critical review of security policy and its associated parts*
- *To illustrate an overview of security requirements and their associated features*
- *To focus on security and access control models*
- *To provide security goals and threats*

3.1 Introduction

Security is a continually evolving process in order to accommodate the latest security concepts. This research introduces some approaches to develop an adequate security system to meet the necessary specifications for utilisation of information technologies (ITs).

Computer systems play an increasing and essential role in our society; therefore, security has become a dynamic and critical issue in IT. Security issues are a fascinating topic for research, which is driven by the promised development of shared networks and their resources. They cover many different areas e.g. physical, networks, platforms and applications. Each of these areas has its own risks, threats and heterogeneous solutions. Security therefore, has become a coherent and crucial issue to protect assets whenever possible. When security is discussed, the theme is usually about hackers and software vulnerabilities. Security has a wide base that touches upon several different areas. Thus, security poses a complex matter for many companies. Developers and designers of security need to comprehend the vision in order to understand how it will influence the design of the system as so many areas of security impinge on each other.

The primary purpose of this research is to develop a high-level security policy consisting of specifications that can be diffused into it by mapping out the primary relationships between services/states and users in a Web services environment. The significant task here is to provide a critical view and formulate security policies that allow the system to be used without breaching security.

3.2 Critical review of Security policy

A critical review is an evaluation and consideration of a research or subject; it makes judgments, positive or negative, based on various criteria. The information and knowledge in the research needs to be evaluated, and the criteria that should be used can vary depending on discipline.

The overall system was deemed a success in that in functioned according to the desired objectives and approaches. The functionality, usability testing concluded that the service behaviour was able to be collected and then evaluated. The validation of this approach is used by various tools e.g. FSM and AGG. FSM supports a lot of modelling that makes programmer and designer to understand the requirement. This approach has examined and implemented by Sun Java computer programming language.

The overall aim of the research was to propose and develop architecture-based observation that uses paramount technology to observe the interaction between services.

A genuine lack of knowledge at the beginning of the research had to be overcome fast; this knowledge has been gathered from many resources. Hence, a slight criticism of this research is that, this system needs to be performed under environment or confederation system. The usability of this approach requires criteria that based on pre-defined services, observer and their associated parts.

3.3 Security Requirements

Security is defined as a set of features and services that tackle a set of security requirements by handling a set of cases. It plays a critical role in the development and functionality of many large scale distributed software systems. Information security is a serious requirement, which must be considered carefully and not in isolation but as an element that is present in all stages of the software development. The importance of security to enterprises and organisations has risen rapidly and it may be attributed to several important trends. Security (Peri 1996) is an integral element of a management system. The main idea of security is to protect our system's valuable resources against an intruder. Through the selection and application of appropriate safeguards, security supports and helps our model's mission by protecting its resources. When a system's role is set and defined, the security requirements implicit in that role can be defined. Security can then be explicitly stated in terms of the organisation's mission.

Security requirements (Bishop 2002; Siewe 2005) are vital to overcome different attacks and threats and are used by different enterprises and organisations to keep their systems safe. Security requirements are concerned with the protection of assets from threats. They state constraints about who is authorised to access resources and information. Moreover, when these requirements are provided and managed, security will be much more easily achieved (Edward 1994; Siewe 2005). Security rests primarily on confidentiality, integrity and availability. The explanations for these three features vary, as do the contexts in which they arise. The explanation of a phase in a given environment is dictated by the needs of the individuals, customs and regulations of the particular organisation (Peri 1996; Bishop 2002).

The definitions most often proposed for security requirement and information security identify three primary requirements: confidentiality, integrity and availability. These requirements should be used and applied in a system in order to keep a high level of security. Confidentiality ensures that the information or resources in a computer's system will be disclosed only to authorised parties. Integrity refers to the protection of information against improper or unauthorised modifications. It maintains and keeps the value and the state of the information so protecting it from unauthorised modification.

A major objective of information security policies is to ensure that information is not modified, destroyed or subverted in any way. Availability ensures that information and information systems are accessible and operational when requested. Furthermore, it is concerned with the facility to use the information or resource that is desired. It is an important aspect of reliability as well as of system design because an unavailable system and information is at least as bad as no system at all (Edward 1994; Siewe 2005). To enforce the above requirements, three mutually supportive technologies are used: authentication, access control and auditing. Authentication is essential to ensure that both end peers are genuine and not impersonators. Without correct authentication, no other requirements can be implemented. In addition, an adversary might masquerade within the operation of other nodes in the network. Authentication is the procedure used to verify the digital identity of the sender of a communication, such as, a request to log in. The sender who is authenticated may be anyone using a computer, a computer itself or a computer programme. Furthermore, authentication deals with the identification of users. Access control means the user has right of access to the resources. This feature is also concerned with limiting the activity of users (who have successfully gained entry into the system) by ensuring that every access to information or resource is controlled and that only authorised accesses can take place. Auditing is the procedure of recording information viz. user IDs and PWs about access to resources so as to be able to establish responsibilities in the event of a security breach (Siewe 2005).

Security requirements (Firesmith 2003) (Bishop 2002) should be based on an analysis of the assets and services to be defended against the intruders and the security threats from which these assets and services should be protected.

As illustrated in Figure 3.1 below, there are clear relationships between assets and services which are vulnerable to security threats. Thus, security requirements exist that need security mechanisms to counter these threats and thereby protect the assets and services. Ensuring (H. Janicke, F. Siewe et al. 2006) the confidentiality, integrity and availability of information is the key issue in the battle for information superiority and thus is a decisive factor in modern warfare. Security policies and security mechanisms govern access to information and other resources. Their correct specification, i.e. denial of potentially dangerous access and adherence to all established need-to-know requirements is critical. A main purpose of an information security policy must be to ensure that information is always available to support critical business processing when it is needed.

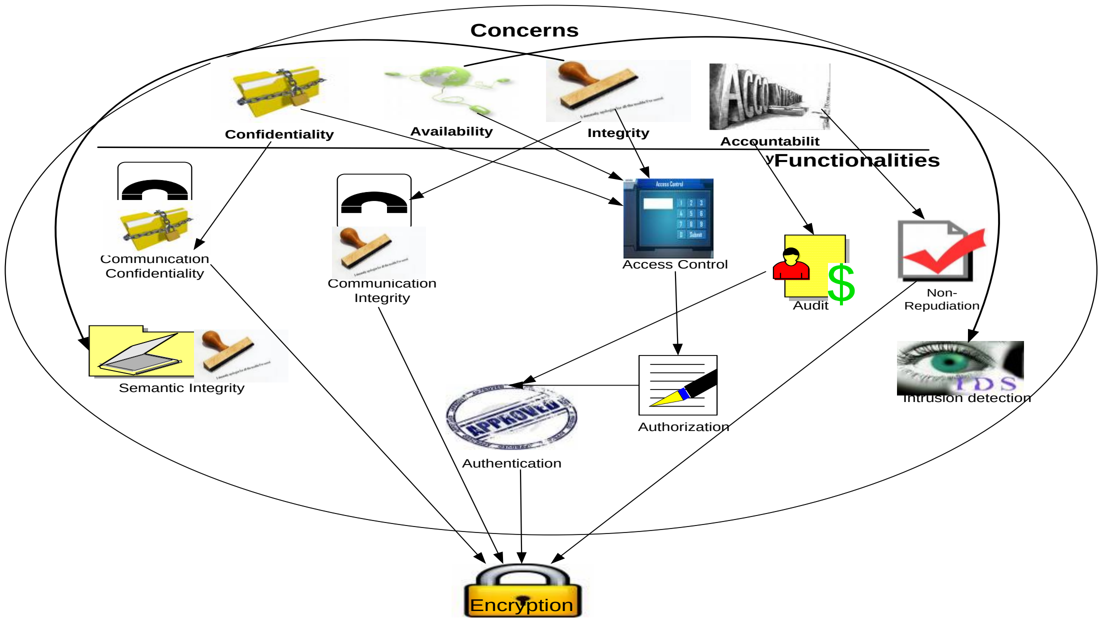


Figure 3.1. Security concern and functionalities

3.4 Goals of Security

Defining security goals is measured by the degree of being secure. The prime purpose of security goals is to present the insights, techniques and methodologies of a system that are used to mitigate threats. A meaningful of security goals are the predefined targeted levels of protection deemed to be adequate. So, in this research we will make long phases of consideration that should show good part of the security goals. And these goals are:

- Prevention

  Prevents attackers from violating security policy

- Detection

  Detects attackers' violations of security policy

- Recovery

  Stops attacks, assesses and repairs damage and continues to function correctly even if the attack succeeds

3.5 Security policy

The rapid growth in security policies has led to the need for a co-ordinating framework and standardisation. Security policy (Harris 2008) represents and expresses exactly what the security level must be by setting the objectives of what the security mechanisms are to accomplish. This essential aspect has a major role in defining the design of the system. A security policy is a base and foundation for the specifications of a system and offers the baseline for evaluating the latter. Security policies should address operating systems and applications of a system. Hence, security policies and their mechanisms manage and govern access to information and other resources (Bishop 2002; Siewe 2005). In addition, security policies are cited as one of the largest cost issues in the maintenance of enterprise networks. Designing and implementing security policy is complex and, in a huge organisation, procedural controls often become vague or cumbersome. Security policies, therefore, have become a popular topic of research, especially in the last few years. They are a high-level specification and requirement of the security of properties and should dominate a given system. Moreover, they are a means for developers and designers to communicate with each other during implementation and validation. Developers and implementers have become increasingly interested in improving security policies which are not flexible and expressive enough to handle the specification and enforcement of multiple policies. In practice, a single policy is not general enough to achieve the level of protection required in many real-world applications; rather a combination of such policies applies.

Therefore, a security policy is a set of collections and specifications of the security requirements of a system. It describes the rules about who is allowed to do what within a system. It (Symon, Qiming et al. 2003) must guarantee the end-to-end agreement for many-to-many inter-operations; ensure the versioning inter-operability and privacy of collaborating partners; and ensure the dynamic establishment of security policies, because any statically defined security policy tends to be unsecured after a certain period of time. It (Symon, Qiming et al. 2003) expresses protection requirements on the system in a precise and unambiguous form. In addition, it describes the requirements and mechanisms for securing the resources and assets between the sharing parties of a business transaction. A modern security policy’s (Rajagopalan 2004) administration practices represent the ability to manage and control network security but some have not kept pace with advances in networking technology. While technologies for designing large-scale networks and network services have advanced dramatically so creating new vulnerabilities and opportunities for difficult attacks, systematic principles for network management, especially security management have lagged behind.

So, security policies can be defined to perform a wide variety of actions with respect to enforcer management (F. García, G. Martínez et al. 2005). There are multiple approaches towards policy specification. These range from formal policy languages that a computer system can directly process, to rule-based policy using some commands or to the representation of policies based on obligation and permissibility rules. To cover this wide range of security policy languages, this research aims to examine the current state of policy engines and their languages by focusing on the approaches enriched with semantics as requirements of policy specification. That will show the strengths and limitations of such languages by comparing policy specifications.

Given the high value and delicacy of security policies, ensuring the correctness of security policies is important but difficult. Any error in security policies directly leads to irreparable damage. To achieve their goals, security policies must undergo systematic and rigorous testing so that they truly represent the intention of their regulation (Vincent C. Hu, Evan Martin et al. 2007; Xie 2008). A major benefit of specifying security policy rules in this way is that a user or an organisation can utilise common aspects that can be shared amongst services and service clients.

3.5.1 Definition of Security Policy

Security policy, heterogeneous in nature must deliver a high level of protection. It declares protection requirements on the system in an exact and unambiguous form. It is concerned with access control, obligations and integrity. It joins the entities in the system and defines the constraints on their interactions (H. Janicke 2007).

In its purest sense, security policy is an explanation of what it means to be secure for a system, organisation or other unit. It defines the security requirements for a given system. It addresses constraints on functions and flow amongst them. It constrains access by external systems and adversaries including applications and access to data by intrusion. Constraints can be imposed by mechanisms like doors, locks, keys and walls (Sattarova Feruza 2008). Since a security policy demands a high-level definition of secure behaviour, it is meaningless to claim an entity is "secure" without knowing what "secure" means. It also makes no sense to consider and address security without tracing the efforts that is required to build a security policy. Another definition (Oriyano 2008) of the term security policy is as a set of obligations that explains how an organisation or a system intends to protect its assets. In another words, it is a collection of declarations of what is permitted and what is not permitted. Alternatively (Karila 1991; Shirey 2000) it is a document that declares in terms of some model regarding a system. It (Shirey 2000) is a set of regulations and practices that specify how a system or organisation grants security services to protect resources and properties. The security policies are factors of security architectures.

Further, it (Karila 1991) is a statement prepared with the information, knowledge and assistance of top management that can act on who/what is…and who/what is not to be authorised to access the area of security during the general operation of the system that is being secured. In other words, security policy is a collection of regulations and practices that states how information is controlled, protected and distributed.

Without a security policy, availability of a system can be compromised. The policy starts with assessing the risk to the network and continues with the need to implement changes in security management practices and for monitoring the network for violations. Ideally, security policies are a set of persistent, immutable and yet practical rules about how systems should be managed and protected (Denis 2006). Consequently, a security policy is a set of specifications of the security requirements for a system. It expresses regulations about who is permitted to do what within a system. In another sense, it describes the goals and elements of an organisation's computer systems.

However, security policies are imposed by organisational policies or security mechanisms, hence, they can be informal or highly mathematical in nature. They are specified by giving a predicate on sets of executions (Fred 2000)and protect and serve as a legal first line of defence against negligence and intrusion.

3.5.2 Significance of Security Policy

A recent spurt in growth of security policy on the Internet has led to conceptualisation of web services. As stated previously, in terms of a meaningful security policy, it is meaningless to argue an entity or service is "secure" without knowing what "secure" means. This urgent matter needs to be resolved. The confidentiality, integrity and availability of digital information are constantly threatened by malevolent software or intrusions. The significance of a security policy has increased dramatically during the past few years and, therefore, it has played an extremely vital role but in an invariably technically fragile business environment. This means secured communication is needed in order for both organisations and customers to benefit from the advancements that the Internet provides to empower their users. Owing to the importance of security, whatever your system, a number of security policies have to be proposed and considered extensively. It is, therefore, crucial to ensure that the whole security policy is enforced by sufficiently robust mechanisms. To guarantee completeness of security policies and assure that they are fully enforced, there are a collection of organised methodologies and risk assessment strategies. In compound systems, such as, information systems management, policies can be decomposed and divided into sub-policies to assist the allocation of security mechanisms to impose sub-policies.

This practice has pitfalls. It is easy to go directly to the sub-policies, which are regulations governing the system’s operations and dispense with the high-level policy. That offers the false sense that the regulations of operation address some overall definition of security when they do not. Regulations of operation declared as "sub-policies" with no "super-policy" usually turn out to be rambling rules that fail to enforce anything with completeness. Hence, a high level of security policy is vital for effective security. Sub-policies and regulations of operation are worthless without it and it is postulated they would be unable to keep states or systems protected.

Web services technology (K. Aldrawiesh, A. Al-Ajlan et al. 2009) as distributed systems are the next innovation in the development of the Internet. The service will allow active objects to become located on Websites so offering distributed services to potential clients. The most crucial benefit of Web services is using machine-to-machine interaction. Web services allow applications to be integrated quicker, more economically and with best effort. Web services standards do not create effective policy. The creation and coordination of security policies are the responsibility and obligation of the participating organisations. As pointed out above, security policies have established a set of security requirements that should serve as the base supported by Web services for building security into Web services applications. A collaborative review (Newcomer 2002) of their security policies by participating organisations can assist to resolve inconsistencies among the various policies and the corresponding sets of security requirements derived from them. Moreover, the collaborative review can itself help to prompt, develop and increase trust in the systems and services controlled by those participating organisations.

In fact, security policies are a powerful resource to assist many enterprises and software applications. Security policies not only state what systems and services are supposed to do but also what actions must be taken when failure occurs. High-quality security practice mandates the creation and periodic review of policies and procedures for reporting, response, and recovery based on the discovery of violations or other security problems (Newcomer 2002). Nonetheless, security policy in Web services or Web computing has become an increasingly complicated matter. It assumes more and more responsibilities and carefulness towards all sectors of the information systems and computer networks. Many applications today, especially in higher education i.e. universities and colleges are based upon interconnecting networks. Although they are used via infrastructure networks and are inexpensive to deploy, therefore, providing security and its associated is complex, especially when dealing with unique types of networks, such as, wireless, Wi-Fi. Hence, providing and managing the security requirements on the Web are a great challenge.

3.5.3 Managing a Security Policy in Web

Comprehensive security policy automation is a relatively active area of research and development. The goal of a security policy is to express some requirements at a high level of abstraction but hiding the details of the implementation that is necessary for their enforcement. The security policy rules are used as the basis for policy specifications. Rule-based languages are well established and well suited because most of these requirements have been previously informally expressed in the form of conditions and consequences. Security policy regulates the rules, requirements and mechanisms for the large Web Services environment. In such an environment, business partners carry out transactions by exchanging Extensible Markup Language (XML) documents encoded in Simple Object Access Protocol (SOAP) messages. The responsibility of the security policy is to define how end-to-end security is enforced. Typically managing and organising a security policy should include the following (Peri 1996):

- How the sender is authenticated, e.g. what mechanism is used, with what parameters and in what range of values?
- Within the SOAP message, which XML elements are encrypted, what kind of algorithm is used, what are the key sizes and for which particular recipient or recipient roles?
- Within the SOAP message, which XML elements are integrity protected, using what mechanisms, and with which algorithms and key sizes?
- In a multiple-hop environment, where to perform security actions, e.g. in building credentials for authentication, authenticating senders, encrypting and decrypting messages, signing and verifying digital signatures.

3.6 Security Models

Designing and implementing secure systems is important, as the number of intrusions has rapidly increased. A security model is a set of structures that offers and gives a policy form and solves security access problems for particular situations. Security models (Harris 2007) integrate the security policy that should be defended and enforced in the system.

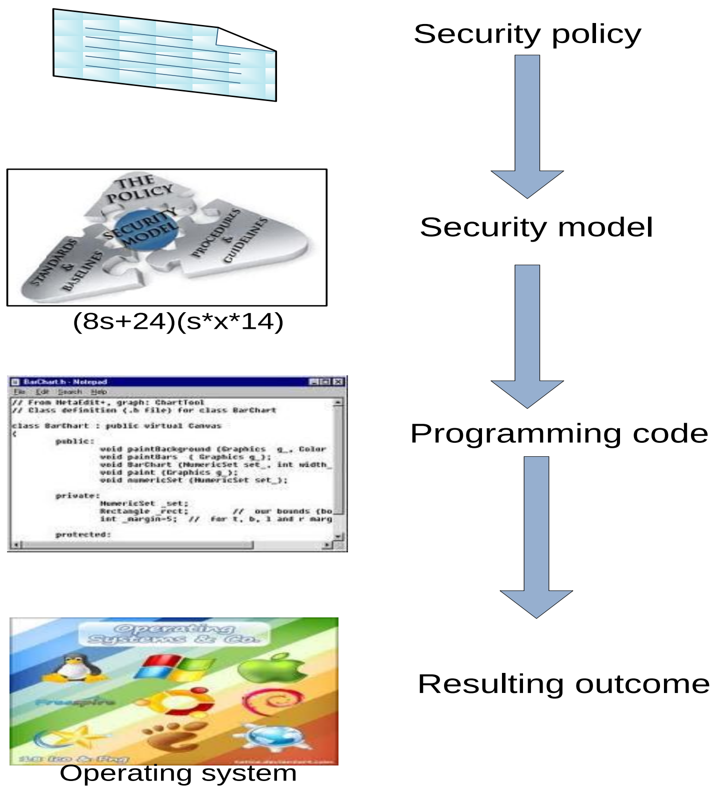
A model is a set of symbolic representations of a policy. It outlines and maps the desires of the policy engine into a set of regulations that a system must follow. A security model states and outlines the abstract goals of the rules to information system by indicating an explicit set of data structures and techniques to defend and enforce the security policy. After that, it has to represent them in sets of maths, formulas and analytical ideas and then outline and map them to system requirements and finally design and develop them by programmers by a computer programming language (Harris 2007).

In very general and simplistic example, if a security policy states that subjects need to be authorised to access objects, the security model could provide and formulate the mathematical relationships and explain how *A* can access *B* only through the outlined specific methods. Requirements are then developed to provide a bridge to what this means in a computing environment and how it maps to components and mechanisms that need to be coded and developed. The programmers afterwards write the programme code e.g. C++, C# to generate the mechanisms that provide a way for a system to use access control lists (ACLs) and offer and give net administrators some degree of control. This mechanism has to show and present to the network administrator a graphics user interface (GUI). This interface enables the administrator to choose via e.g. check boxes, which subjects can access what objects and to be able to adjust this configuration within the operating system. In spite of this rudimentary example, the security model can be very complex and complicated to implement in a system but it is used here to express the relationship between the security policy and the security model. Some security models, for example, the Bell-LaPadula model enforces rules to provide confidentiality protection whereas the Biba model enforces rules to provide integrity protection. Both of these models are used to provide high assurance in security. Informal models, such as, Clark-Wilson are used as a framework to state and describe how security policies should be demonstrated and executed and several security models have been designed to reinforce security policies (Harris 2007).

Further, security models (Peri 1996; Bishop 2002) are used to formalise security policies, in particular to depict the security relevant features of information systems that permit one to reason about their behaviour. Many access control models are defined to carry out different kinds of security policies. These models are primarily used for describing the protection features of systems, e.g. operating systems and application software. Access controls are sets of security features that manage how others and systems communicate with other systems and resources. Further, they protect the contexts, systems and resources from unauthorised access and can be components that participate in affecting and determining the level of authorisation after an authentication procedure has been successfully completed.

3.6.1 Relationship between a Security Policy and a Security Model

As illustrated by figure 3.2 below, the relationship between a security policy and security model is complicated and strong. A security policy sketches and outlines goals without how they will be achieved.



```
// From MetaEdit+, graph: ChartTool
// Class definition (.h file) for class BarChart

class BarChart : public virtual Canvas
{
        public:
                void paintBackground (Graphics  g_, Color
                void paintBars  ( Graphics g_);
                void BarChart (NumericSet set_, int width_
                void paint (Graphics g_);
                void numericSet (NumericSet set_);

        private:
                NumericSet _set;
                Rectangle _rect;        //  our bounds (bo
                int _margin=5;  //  for t, b, l and r marg

        protected:
```

Figure 3.2. Relationship between a Security Policy and a Security Model

There are many types of entities, such as, a user, a programme that requires access to other network entities and resources that are subject to access control. Access control is a broad term that covers several different types of mechanisms that enforce right of entry control features on computer systems, networks and resources. Access control is extremely important because it is one of the first lines of defence in battling unauthorised access to contexts, systems and network resources. When a user is offered or prompted for a username and password, this is called access control. Once the user logs in and later attempts to access a file, that file may have a list of users and groups that have the right to access it. When the user is not on this list, the user is denied entry. The users' permissions and rights could be based on their identity, clearance and/or group membership. The ability that access controls give organisations is to manage, restrict, observe and protect resource availability, integrity and confidentiality. Access controls are one of the foundations of computer security. They are becoming an important issue in research in many fields. It enables and facilitates a user to control access to assets in a given system. In addition, it relates to security features that manage and control who can access resources in the operating system(Peri 1996; Harris 2007).

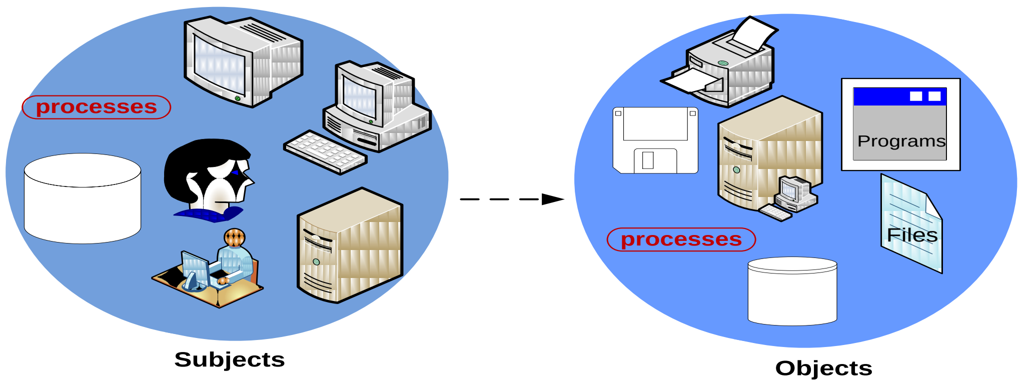


Figure 3.3 Subjects are active entries that access objects, while objects are passive entities.

The Access Control Model, as the name proposes controls access to information. A framework states and dictates how subjects access objects. Furthermore, it applies and uses access control technologies and security mechanisms to enforce the regulations of the system. Access control models are broadly categorised into (H. Janicke 2007): Discretionary Access Control (DAC) and Mandatory Access Control (MAC) Models.

Each model type uses different methods and techniques to control and manage how subjects access objects and each has its own merits and limitations. The business and security goals of an organisation will help prescribe what access control model should apply. It depends on the organisation's needs for conducting a particular type of business. Some organisations use only one model, whereas others combine them to be able to provide and offer the necessary level of protection. These models are built into the kernel of the different operating systems and possibly their supporting applications. Every operating system has a security kernel that enforces a reference monitor concept, which differs depending upon the type of access control model embedded into the system. For every access attempt, before a subject can communicate with an object, the security kernel reviews the regulations of the access control model to determine whether the request is allowed (Harris 2008).

Discretionary Access Control (DAC) is an access policy that is set and determined by the owner of an object. It offers users the means for defining the access control themselves. It sets the restriction to access objects based on the identity of the subjects and groups to which they belong (Siewe 2005; H. Janicke 2007).

Mandatory Access Control (MAC) is an access policy that is set and determined by the system, not the owner. It is used in multi-level systems that process highly sensitive data e.g. military policies. In a mandatory access control the system's security policy is under the control of a dedicated administrator, for instance, MAC policies are Bell-LaPadula or the Chinese Wall Policy. MAC involves centralised mechanisms to control access to objects with a formal authorisation policy (Siewe 2005; H. Janicke 2007).

3.6.2 Other Security Models

As pointed out, security models are a set of declarations that identify the requirements that support and implement a certain security policy. If a security policy states and shows that all users/others are to be recognised, authenticated and authorised before accessing a network's resources, the security model might possess an access control matrix. The matrix should be constructed so that it fulfils the requirements of the security policy. Several security models exist to enforce security policies for example, Bell-LaPadula Model, Biba Integrity Model Graham-Denning Model, Harrison-Ruzzo-Ulman Model and Brewer and Nash Model, in order to emphasise those aspects of their policy models (Bishop 2002; Siewe 2005; H. Janicke 2007; Harris 2008).

3.6.2.1 Event-Condition-Action (ECA)

ECA model is a powerful paradigm for programming reactive systems. The fundamental construct of ECA model is reactive rules of the form On *Event* If *Condition* Do *Action* which means: Event occurs, if Condition is satisfied, then run Action. ECA systems receive inputs (mainly in the form of events) from the external environment and reacts by performing actions that change the stored information (internal actions) or influence the environment itself (external actions). ECA has to satisfy many properties to be useful and reached in a wide range spectrum of applications. First of all, events occurring in a reactive rule can be complex, resulting from the occurrence of several basic ones. A widely used way for identifying complex events is to rely on some event algebra, e.g. to introduce operators that define complex events as the result of compositions of more basic ones that occur at the same or at different instants. Actions that are triggered by reactive rules may also be complex operations involving several (basic) actions that have to be performed concurrently or in a given order and under certain conditions. The possibility to define events and actions in a compositional way (in terms of sub-events and sub-actions), allows a simpler programming style by dividing complex definitions into simpler ones and by permitting the use of the definition of the same entity in different fragments of code (Alferes, Banti et al. 2006).

3.6.2.2 The Summary of Security Models

Security models have become one of the most of active topics in the range of security technology. They have been built to support the requirements of their policies. As a result, a summary of most security models is listed below in table 3.1 with their job description:

Model	Description
DAC	Determined by the owner of an object
MAC	Determined by the system, not the owner
RBAC(Non DAC)	Relates to the concept of roles within the subjects and objects

Access matrix	A table of subjects and objects indicating what actions individual subjects can take on individual objects
Bell-LaPadula	Protects the confidentiality of the information within a system
Biba	Protects the integrity of the information within a system
Clark-Wilson	Protect the integrity of data and ensures that properly formatted transactions take place
Brewer and Nash	Allows for dynamically changing access controls that protect against conflicts of interest
Graham-Denning	Creates rights for subjects, which correlate to the operations that can be executed on objects
Harrison-Ruzzo-Ullman	Allows access rights to be changed and specifies how subjects and objects should be created and deleted
ECA	On Event If Condition Do Action

Table 3.1 a summary of most security models

3.7 Security Mechanism

A mechanism may be procedural, technical or physical. A security mechanism (Bishop 2002) is a unit or procedure that enforces some stages of the security policy. It is a scheme, tool, process or procedure to enforce a security policy. It is an algorithm or logic that applies a particular security enforcing or security relevant function to any hardware or software. Alternatively, it may be considered an entity that enforces some part of the security policy. It is used to implement security services.

In general, developers and designers of computer system have achieved many things such as, where security mechanisms should be located and performed i.e. in software, hardware, kernels, operating systems or services and identified the objects that are included and how those components should interact with each other. A security perimeter (Harris 2008) has been formed that isolates and separates trusted from the distrusted components. In addition, proper interfaces have been developed for these entities to communicate securely. The current need is to build, improve and apply a mechanism that ensures that subjects that access objects have the necessary permissions to do so. This means the designers should enhance and perform a reference monitor and security kernel. In most models, the security mechanism works under several assumptions:

- Each mechanism is designed to apply one or more parts of the security policy.
- The union of the mechanisms applies all features of the security policy.
- The mechanisms are applied correctly.
- The mechanisms are installed and administered correctly.

Therefore, “A security mechanism is defined as the low level (software and hardware) functions that implement the controls imposed by the policy and formally stated in the model.” Furthermore, it is a method, tool or procedure for enforcing a security policy. It detects and prevents attacks and recovers from those that succeed. Analysing the security of a system requires an understanding of the mechanism that enforces its security policy (Siewe 2005). The figure 3.4 below shows the sequence of security heterogeneous.

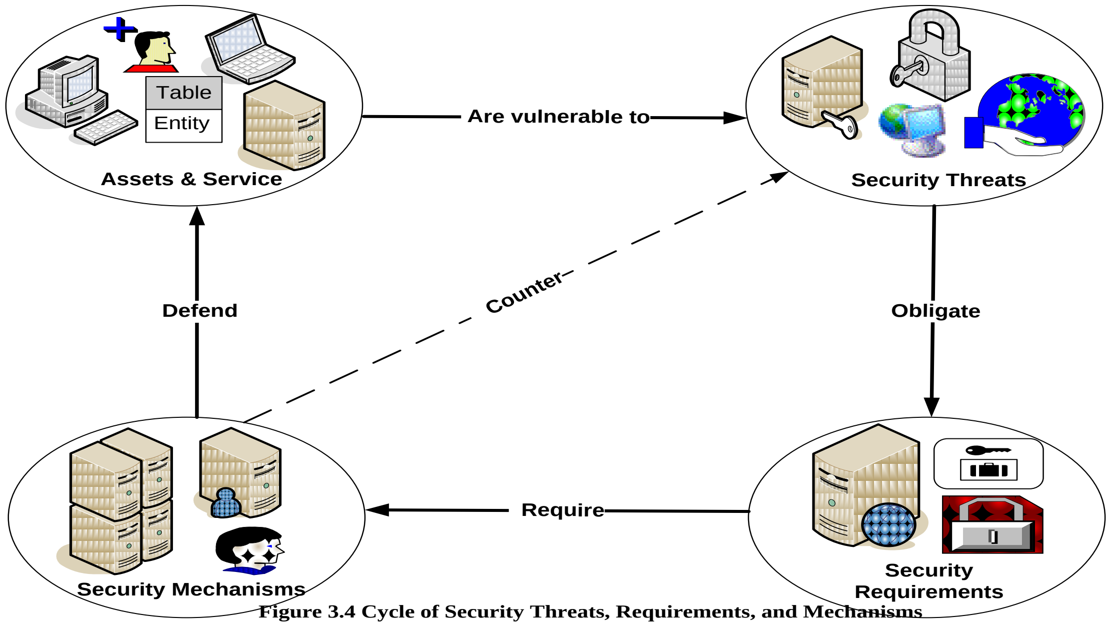


Figure 3.4 Cycle of Security Threats, Requirements, and Mechanisms

The prime goal of a security policy is to define the means for facing a given context of threats. Having and managing a model leads to the definition of some important rules that could be taken to support formalising and designing security policy requirements for organisations with a particular high degree of security (Shimonski 2003).

3.8 Security Threats

Increased usage of the Internet has resulted in the growth of security attacks against users and systems. Attacks are increasing as the value of the data increases. This means that an organisation must come to estimate the value of the data it holds. An ever-increasing amount of information and data are needed for many purposes e.g., education and business and with the advent of the Internet threats to these values are increasing apace. Techniques are devised to protect systems, services and contexts from known threats. Security attacks cannot be deflected from threats that are either unknown or have no universal definition. The high level of definition and assurance in security properties of systems used in security-critical areas viz. finance, military and education is usually accomplished by verification. In addition, a key role to avoid security threats is to realise and identify the IT infrastructure's vulnerabilities and to take corrective action. When corrective action is complicated, attacks must be observed to assess and estimate the time and scope of penetration by perpetrators. One, therefore, needs to develop a technique for carrying out this verification in a systematic manner and this will be discussed in chapter 6.

3.8.1 Definition of Threats

The principal function of a security threat is to classify and define atypical security activity within a system. In general, a threat is a set of circumstances that have the potential to cause loss or harm to something of value. It (Edward 1994) is in a computer system that any potential occurrence and maliciousness can have an undesirable effect on the properties and resources associated with the contexts. It is a potential violation of security activity. When a violation occurs, those actions are called attacks. Those who cause or defend such actions are called attackers. The following list includes a brief description of some common threats (Siewe 2005; Harold F. Tipton and Krause 2009):

- **Phishing:** In general, phishing is carried out by email when a user/receiver is instructed to click on a link/button that takes them to a counterfeit website that could request their personal information.

- **Hackers:** The term 'hacker' can infer computer proficiency. The term describes a skilled computer user who uses this knowledge and information to achieve an advantage over those who are less familiar with computing technology. The ability of the hacker is used to facilitate and perpetrate cyber crime.

- **Bots:** The term 'bots' refers to a computer that can be remotely accessed and controlled by software in conjunction with thousands of other computers that have been compromised in the same fashion. Bots are designed and created by malware that allows an unauthorised individual remote access to a network computer. This type of malware is known as a ‘backdoor’.

- **Password Cracking**: Is a process that recovers passwords from data transmitted by a computer system. Passwords are the most common method and technique of authentication used to control access to digital resources viz. data or information. They are also the easiest way to accomplish unauthorised access to these resources. Armed with password cracking software, an intruder can break a dictionary word password in seconds. When setting a password a user should consider how much information is saved and solely protected by it. It quickly becomes clear that good passwords are vital to preserving confidentiality.

- **Malware:** Refers to any computer programme (software) that executes without the full knowledge and consent of the system’s owner. There are several types of malware e.g. bots, backdoors, rootkits and spyware. Malware usually makes computers operate more slowly and may damage files and hard drives.

3.8.2 Types of Threats

Threats have remained a major challenge to computer security for a long time. The threats that a site faces and the level and quality of its countermeasures is based on the excellence of its security services and supporting procedures. The specific mix of these attributes is governed by the site's security policy, which is created after a careful analysis of the value of the resources on or controlled by the system and of the risks involved.

Essentially, there are two types of threats (Edward 1994): Behavioural and Software. Often, these threats are used in combination. Behavioural threats, more commonly known as social engineering, are designed to obtain and install malicious software (malware) or to reveal personal information. Malware generally offers and provides unauthorised access to the use of the computer. Revealing personal information, i.e. account numbers can lead to fraud and identity theft. Listed below are brief descriptions of common social engineering techniques and different types of malware.

- ✓ **Disclosure:** Dissemination of information to an individual for whom that information should not be seen
- ✓ **Modification:** Unauthorised change to information stored on a computer system or in transit between computer systems
- ✓ **Denial of service:** Access to some computer system resource is intentionally blocked as a result of malicious action taken by another user (Edward 1994)

3.9 Summary

This chapter has provided an introductory tutorial on the importance of security models, services and mechanisms. We have introduced and presented an extended account of related work on security policy with their associated features and some related concepts and techniques, for discovering and expressing security policies.

Several proposals of security models have attempted a combination of mandatory flow control and discretionary authorisations. Security models are a collection of structures that present features of a policy and support security access problems for particular situations. They have become one of the most important topics as specified by the large number of researchers that have been initiated in recent years. These researchers' efforts have led software vendors to integrate model requirements into their products. Security models set a number of security issues, and these issues are covered by the system’s security policy. All these security models cover a wide range of security policies encountered in practise.

In this chapter, we have determined the importance of using the security policies in order to protect the resources from threats.

Security policy is still a work in progress both from the development of emerging standards and from the security community’s growing awareness of the benefits and pitfalls of this new integration paradigm.

CHAPTER 4

ARCHITECTURE

Objectives

- *To define the architecture of Observation and its associated components*
- *To classify the observation and its associated components*
- *To provide an enhanced observer model and its advantage and disadvantage*
- *To present evaluator and enforcer models and their associated parts*

4.1 Introduction

Administrators have to take different networking technologies and distributed applications into consideration in order to control them. The challenge of controlling such vast information technology is rapidly becoming complex. The growth of these heterogeneous systems and technologies has led to their becoming more complicated, which encourages security violations to occur with increasing frequency. This makes monitoring information technology systems a day-by-day necessity. Consequently, the ability of observation systems must increase. There is an increasing need to ensure the security of observation systems plays a critical role in technical and theoretical impact. The complexity of security of observation systems is growing as the system's architecture and application has to fulfil the requirements of ever demanding project scenarios. Due to the growth in the complexity of today's software systems, new ideas for the design and management of those systems have to be discovered. Most software designers and programmers still design their systems following the top-down approach, trying to control this increasing challenge. As the numbers of resources that need to be managed and controlled grow, the task of managing these devices and applications depends on numerous system and vendor specific issues. To avoid the operators drowning in excessive detail, the level of abstraction needs to be raised in order to hide system and network specifics.

In general, the observer always observes everything that can be seen within a context and the result should be tackled and passed to a superior. Observation can be an activity of a living being e.g. a human via the reception of information from the environment through recording data using a set of scientific instruments. In addition, Observation is a collection of qualitative research methods and, therefore, has previously recognised strengths and weaknesses. It can be used quantitatively, for instance, counting the number of states. It may be a diagnostic tool to help and understand what is occurring. Observation can also produce descriptions, such as, number, time and duration of events. It can reduce risk and emphasise meanings and experiences (K. Aldrawiesh, Siewe et al. 2011). It is a set of research methods that is qualitative in nature and offers information that can help and support by highlighting a problem. The observation's methods include open or structured observation which may be overt or covert. The 'observation' is informally divided into two categories as follows:

- Unfocussed observation
- Structured observation and this category are based on qualitative.

The observation is defined as a science class of research that encourages and supports the development of a system for an administrator's observational abilities central to the scientific method by integrating, writing and recording the component events. The observation is a useful approach for developers and researchers. It allows the analysis and differentiation of recent developments in order to discover the damage. Also, it categorises the various researches by classifying the elements that are essential for building an observation system as well as discussing the necessity of new observation and control structures in Web context systems, starting from the basic contradiction between bottom-up behaviour and top-down design. Observation systems serve the purpose of keeping emergent behaviour within predefined limits.

4.2 The framework of Architecture

As shown in figure 4.1 below, the architecture consists of many components that are designed to be used for several purposes. This architecture identifies common environment syntax for expressing functional or non-functional properties of a Web service in a declarative manner. In addition, it illustrates description and communication between the components and their structures, which describes a broad range of service requirements, preferences, and capabilities.

Moreover, this architecture includes a solution of optimal entities that have connected to control service behaviour whenever interactions and communication have occurred. These entities are commonly used in observation systems, such as, policy, e-observer, evaluator and enforcer. The aim here is to provide a formal framework in which observation systems can be expressed together with their security requirements at different levels of abstraction. Secure in this context means the system presents adequate mechanisms to enforce dynamically changing security policies that define constraints on the system’s behaviour.

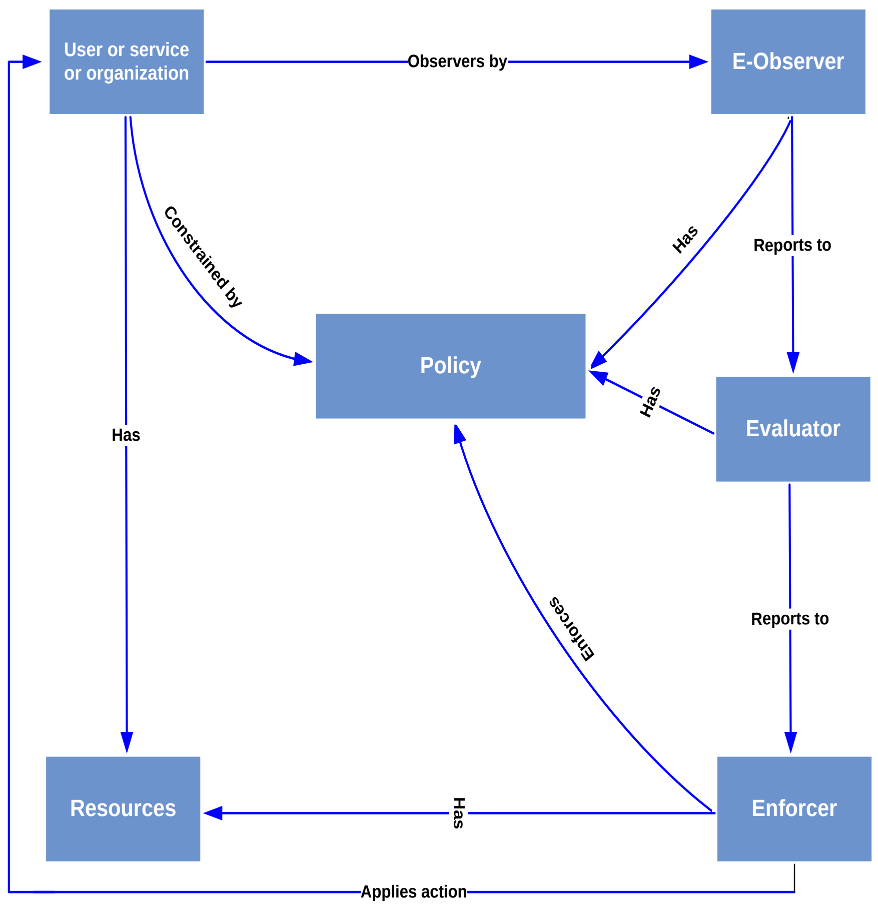


Figure 4.1 Framework of Architecture

4.2.1 Policy:

The main part of this architecture is policy. The policy is constrained by the behaviour of system components. The policy (Andrew, Geoff et al. 1994; Raju Rajan, Dinesh Verma et al. 1999) becomes an increasingly popular approach to the dynamic adjustability of applications and management. Policies are often requested to be applied to automated network administration tasks, such as, configuration, security and QoS. Furthermore, policies can manage and control the system's behaviour, as they become a key to permit those who can access a system. In our observation model, policy manages and controls the system's behaviour. It works as a key to decide who can access information or data. Other parts of this architecture have been described in chapter two.

4.3 Flowchart of the Architecture

Figure 4.2, below, shows flowchart of the architecture which is proposed and drawn based on the essential architecture. It provides the crucial parts of the sequential stages, beginning with a user or customer, which is constrained by global policy. This policy obligates these properties (services) such as, the e-observer, the evaluator and the enforcer/actuator. Each one has a different policy depending on their job. In addition, each one of them has a local policy also regarding their job. Figure 4.2 shows the sequence of the data flow in the model to demonstrate how the system could be working to control any violation activity.

Firstly, services will be collected into a container and when they communicate with each other the e-observer will start observing the service behaviour based on its security requirement. Then whatever is observed should go to the evaluator which will start to evaluate this observation. If the evaluation is satisfied then it can pass to the end, otherwise the evaluation is not satisfied and needs to be enforced by the enforcer.

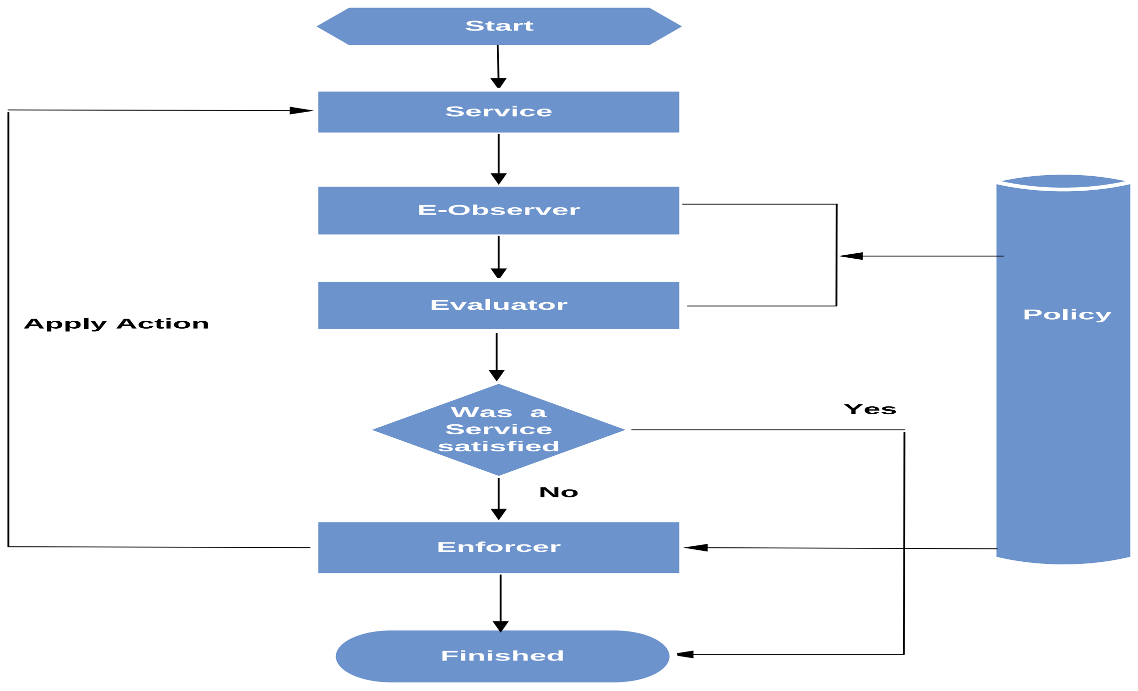


Figure 4.2 flowchart oF the Architecture

4.4 Observation Model

Observation systems are vital, meaningful for safety and need to be highly dependable. They are a collection of components that connect and adhere service behaviour, whenever interactions occur, with the expected results of these interactions by making an aggregation of the results to accomplish a satisfactory System Observer Model (SOM) (K. Aldrawiesh, Siewe et al. 2011).

Observation systems have evolved and grown significantly over the years and are becoming an essential active tool for different organisations and their security applications. They are traditionally used to enhance the security of a system. Furthermore, they are found and used in several enterprises for many purposes, for example, education, business and finance.

Figure 4.3 below shows the observation framework which includes components that are used to manage the sequence of the service. A closed loop control consists of controlled process sensors. These components can be typically found in security systems that are used for observation. Our observation framework consists of three main components (K. Aldrawiesh, Siewe et al.):

- ✓ Enhanced observer model
- ✓ Evaluator model
- ✓ Enforcer model

In addition, system observer model (SOM) is a collection of services that are designed to receive many services.

Observation technique is an important issue and adopted by different companies to enhance performance and control their ability to examine in close detail, service behaviour. Moreover, observational technique is a process and method by which an individual gathers firsthand data about programmes, processes or behaviour that are being investigated. It facilitates an opportunity to collect information on a wide range of behaviour, to monitor a great variety of interactions and to explore openly the model's requirements. By directly observing operations and activities, the e-observer can develop a holistic perspective, such as, understanding the context within which the project operates.

As mentioned, our model includes an e-observer which maintains a list of its dependents and then automatically reports any changes in state of service to the evaluator model by calling one of their methods.

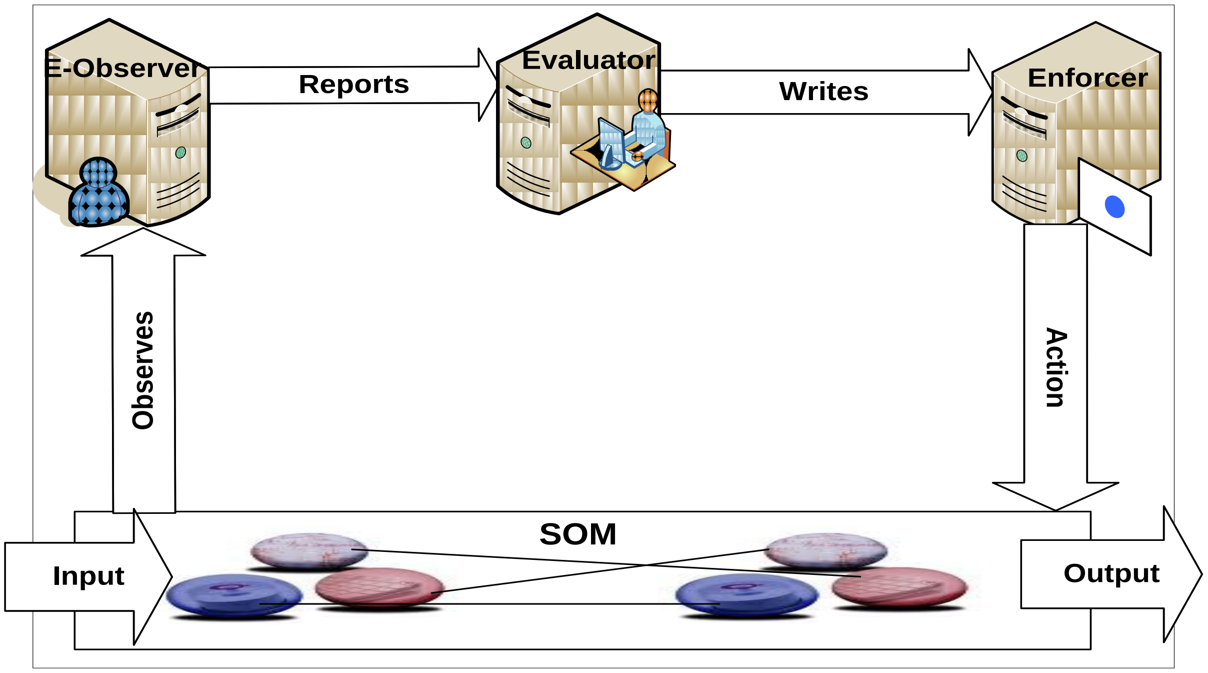


Figure 4.3 Observation Model

Moreover, e-observer techniques (Jürgen Branke, Moez et al. 2006) address most problems and provide and govern a proven solution that is re-usable in similar contexts. The proof of the solution lies in its ability to provide a facility for the system and other associated parties to collaboratively perform and control any violation activities. It addresses these activities by a technological approach. Figure 4.3 states that the e-observer is responsible for appropriate observation and feedback. The sensor and actuator are at the heart of SOM. They are devices that measure a measurable attribute and convert it into a signal which can be read by an observer. A control-closed loop is created on top of the SOM to rotate the states/services. Thus, the e-observer observes behaviour through sensors by comparing the results with the expectations and decides what action is necessary and directs the SOM to provide the best-known action through the enforcer system, but only after the evaluator has assessed the results.

We have to pre-suppose some criteria for use when observing services in a system, comparing the expected knowledge of historical and current data and the rules that are assumed. The e-observer's task is measured in order to quantify and predict emergent behaviour with the basic metrics of QoS e.g. time, availability and security.

Despite the good progress in terms of the security that is being made by such observation systems, only a few results have been reported in the literature, in particular, Web services. Security in observation is an important issue that needs to be explored. This research deals with e-observer techniques over the Web environment, which makes it possible to achieve adaptability in order to improve system performance in dynamic environments by efficiently using the available resources and controlling the policy's model.

In spite of presenting security in previous chapters, the observation model has to address and animate the security to protect its resources. The prime purpose of security here is to present the insights and techniques of the model that can be used to mitigate threats.

4.4.1 Enhanced Observer Model

As seen in figure 4.4, the e-observer collects and gathers information (raw data) from the SOM and then the aggregated values are reported to the processor. Subsequently the evaluator model assesses the situation and feeds back the appropriate action to the enforcer model to influence the SOM. The raw data which is collected from SOM is not unprocessed.

The observation behaviour itself is variable; hence, the e-observer model influences the observation procedure, e.g. by selecting certain detectors or certain attributes of interest. The feedback from the e-observer to the evaluator directs attention to certain observables of interest in the current context.

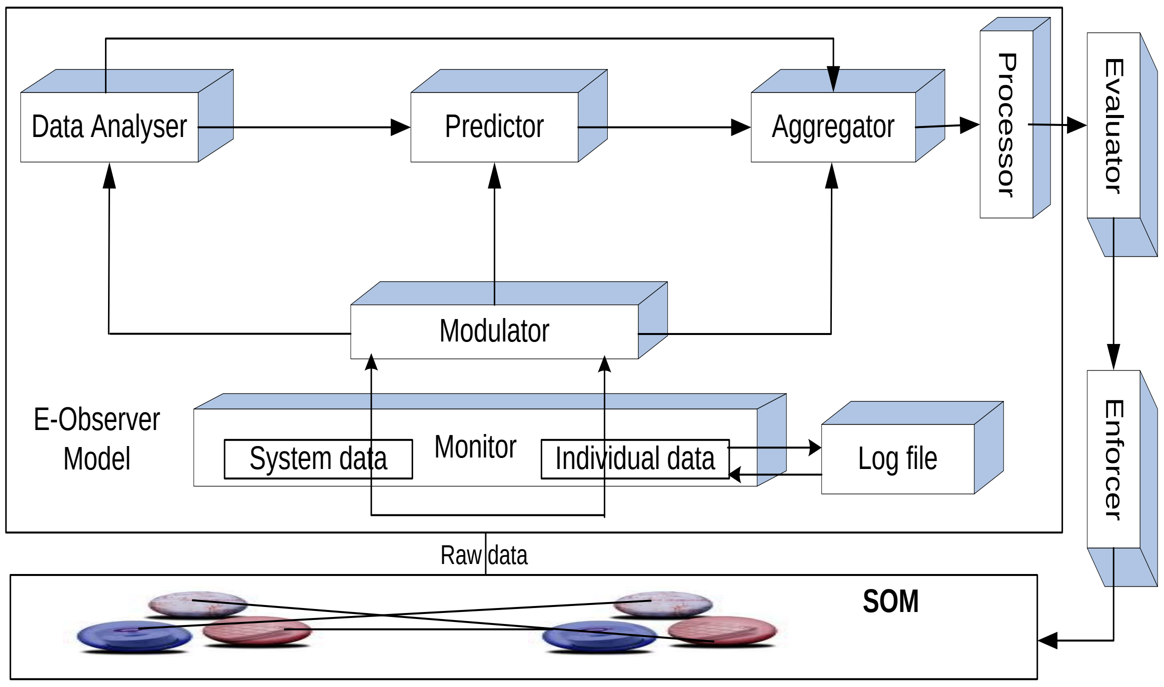


Figure 4.4 Enhanced Observer Model

Based on the aggregate results by the e-observer, the evaluator can benchmark the data with an objective function and knows what actions are best to guide the SOM in the desired direction by informing the enforcer model, which acts as a switcher to detect the violation (K. Aldrawiesh, F. Siewe et al. 2011). Our assumption here is, tracing and observing the communication between services and then taking the decision based on their behaviour and history. Hence, the big issue here is how do we ensure that some given security requirements are satisfied and enforced? The scenario in our model is based on the following:

- System’s components are Web-services.
- These components are black boxes and designed/built by various vendors.
- Topology is highly changeable

The "observation" model allows to observe (partial) of behaviours of those services and to construct complete behaviours that can be analysed and compared with security requirements. So we have two sets of behaviours: (a) the observed (partial) behaviours; and (b) the actual required behaviours.

These two sets can then be compared and determined to see if the system satisfies the security requirements. If the two sets are the same then the system satisfies the requirements; otherwise it does not.

Therefore, the e-observer observes these services when they interact with each other and then the evaluator model assesses the communication between them to discover any deviation that is based on their behaviour and history. The evaluator then writes to the enforcer model for detection. In a technical system, the e-observer plays the role of the limbic system. It observes the external environment via the sensory input as well as the internal behaviour of the low-level execution unit and is highly manipulative in several ways. The usage of the e-observer structures can be regarded as the introduction of emotions into technical systems.

4.4.2 Definition of e-observer

The e-observer is a set of structures that collects information about service behaviour by tracking it to ensure it acts in a suitable way. It provides feedback that is superior to the evaluator model. It works by combining knowledge about the information of services, such as, behaviour, parameters, attributes and feedback to extract action that is better than can be obtained and detected.

In addition, e-observers are a worthwhile field for research and development. They are considered promising methods for the development of shared heterogeneous systems. The principle used by the e-observer is to combine a measured feedback with knowledge of the System Observer Model’s (SOM) components, which is reported to an evaluator model via many operations. The e-observer is used to enhance system performance and security. It can be more accurate than the phase lag inherent in the sensor. Moreover, it studies a group of subjects engaged in certain activities but does not directly participate in them.

E-observer technology is not a panacea; it could work with assistant tools by adding complexity to the system. This will require computational resources with a high cost. In the meantime, observation as a technique is required by several enterprises to enhance and control their ability to examine, in detail, service behaviour. E-observers are used in many systems in different ways with high efficiency and good results. They are simply listeners that have been used for years by selected industries and they use complex mathematics. In our model, they are defined as digital algorithms that combine outputs with knowledge of the SOM to provide results superior to traditional structures that rely wholly on the evaluator via the processor.

Programmatically an e-observer is a software design in which an object, called the subject, maintains a set of observable dependents and notifies them automatically of any state changes by calling one of their methods. The e-observer works as a listener by registering their interest viz. service with observables that alert all registered listeners when an update is required. Consequently, it observes and analyses the state of the system by checking and comparing the correctness of the observed state of the system with its expectation.

4.4.3 E-observer's Technique

The e-observer is able to add and remove observables to or from observers. It ensures that updates are performed correctly. In addition, the e-observer class recursively updates as registered observers while avoiding multiple updates and cyclical behaviour. Figure 4.5 below shows methods to add and remove objects:

```
 void Register(Object anObservable, Object anObserver)
     If anObservable is registered with the manager, add anObserver to the list of observers for
      anObservable,
   otherwise, register anObservable and add anObserver to its list of observers.

void Deregister (Object anObject) removes all references to anObject from the manager where
   anObject is an observer, observable, or possibly both of these.

void Deregister (Object anObservable, Object anObserver)
   Remove anObserver from the list of observers for anObservable. If anObserver is the only observer,
the observable object anObservable is also removed.
```

Figure 4.5 Observer can add or remove objects

The e-observer updates and notifies all registered observers of a particular observable when an update is required. Where an observer is also an observable the method recursively processes all of its observables and their dependencies. Multiple updates and cycles are avoided by viewing the update. A process is a graph traversal, which maintains a list of visited objects. An argument parameter and a reference to the observable implement both the push and pull models of event notification. Figure 4.6 below refers to the method to update all registered observers.

```
  void notifyManager (Object anObservable, Object arg)
  {
    if ( anObservable s exists )
    { for ( all observers of anObservable ) // process in reverse order of registration
    { currentObserver.notify (anObservable, arg) // combine push and pull models
    place the current observer on the visited list
  if ( currentObserver is also an Observable )
call notifyManager (currentObserver, arg)
          }
     }
 }
```

Figure 4.6 Methods can update all registered observers

Logically, an e-observer system is not a sufficient solution; it needs technical tools to work perfectly. It is also used to regulate an enormous variety of services, software and processes. It controls and manages the state attributes, parameters and interactions of service behaviour. It is common to use the concept of e-observer techniques to address certain problems relating to application design and architecture. The definition of an e-observer is often difficult to convey with any level of accuracy which the concept warrants, so a proposal of observation is used as the conventional means.

4.4.4 Duties of the e-observer

The e-observer is a piece of a system that monitors and catches an interaction and execution of the communication of the services behaviour and meets properties in accordance with its policy. Observing allows a tool or user to monitor and analyse to recover detected faults and to provide the information to the evaluator. Thus, the e-observer is a component of the software or hardware that observes interactions and events of the service behaviour or any interactions between services. It, however, does not participate in these interactions.

In addition, it is a technique used to confirm that the security practices and controls in place are being amenable and effective. It also supervises the interception of communications, the interacting checking of our system, the logging, recording, inspecting and auditing of data.

As a technical definition, the e-observer pattern is used to keep consistency between related objects while minimising their coupling and maximising re-usability of the objects. Therefore, the e-observer observes the service behaviour of the SOM in terms of system parameters that are well defined.

In general, the e-observer is advantageous in creating a clear distinction between various objects in the system. It is also common to find this solution utilised within a non-user interface (UI) related segments of a framework or application. The usefulness of the e-observer pattern extends far beyond its original intent.

In addition, the e-observer has some factors and duties that should organise its capability and its observation of the framework. The usage of e-observer structures can be regarded as the introduction of emotions into technical systems. The overall objective and challenge of observation is how to guarantee the e-observer captures suspicious services and to ascertain how effectively they can be contacted and implemented within a web service environment or decentralised system.

Having and managing services and their associated components on the Web is a meaningful and significant challenge regarding their dynamic and unpredictable nature. To ease this potentially dangerous situation, an observation model tries to mimic key phenomena when interactions occur between services in order to obtain what actions should be requested. Therefore, the duties of e-observer are as follows:

- Record available information of an event, effort and estimate total catch
- Collect critical information e.g. history, behaviour and risk preference
- Monitor services to determine the level of fulfilment of policy’s regulations.
- Record incidental takes and interactions of events of services

4.4.5 Advantages of the e-observer

The e-observer has some advantages that may enhance the prospect of optimal solution for the security:

- ✓ Minimal coupling between subject and observer
- ✓ Provide direct information about behaviour of services
- ✓ Permit/report to the evaluator to enter into and understand situation/context
- ✓ Provide good opportunities for identifying unanticipated outcomes
- ✓ Exist in a natural, unstructured and flexible setting
- ✓ The observer can add new independent observers
- ✓ Observer does not participate with objects that have been observed

4.4.6 Drawbacks of the e-observer

The principal difficulties in using observation are the large amount of information. In spite of the advantages of the e-observer, some drawbacks need to be considered, such as,

- ✓ Expensive and time consuming, which leads to depletion
- ✓ May affect the behaviour of participants
- ✓ Selective perception of observer may distort data
- ✓ The investigator has little control over the situation
- ✓ The behaviour or the set of behaviours observed may be atypical

As abovementioned in figure 4.4, the e-observer is the main component of the observation model that aggregates available information with complex calculation of QoS metrics then reports to the evaluator model via the processor. The e-observer model has some components, as follows. (Eales **2005**):

4.4.7 E-observer model’s components

As mentioned in figure 4.4, the e-observer model has many components as follows:

4.4.7.1 The Monitor:

The monitor is an explorer tool that has needs to verify and watch states/services while in operation. It processes the data coming via the SOM. As previously pointed out, SOM has a control loop to rotate the states/services. The SOM is considered a set of factors possessing certain attributes. The monitor samples the attributes of the SOM according to a sampling frequency given by the observation framework and its policy. The information coming from the SOM constitutes as raw data (unprocessed) for the e-observer, which can be classified into individual data common to all elements of the system and some global system attributes. In order of occurrence, monitoring the SOM is nothing else than the generation of a time series that reflects the current state of the system as well as its history. The sensory equipment of the SOM limits the selection of observable attributes and the resolution of the measurement.

4.4.7.2 The Log file:

The log file is a process that includes a record of events/actions that have occurred. It is generated automatically by some applications and is typically stored as text editable files. As a sign of the significance of the data, in every loop of observing the SOM, all measurable data is stored in a log file. This stored data can be used within the predictor for the calculation of time-space-patterns by the data analyser.

4.4.7.3 The Modulator

The modulator is a co-ordinator process component that is located between the predictor, data analyser and monitor. The modulator is a process that facilitates the transfer of information to another. It is the process of varying one or more of elements and attributes of high value. It conveys information/messages to a superior. In the next step, some derived attributes can be computed from the raw data, e.g. an attribute velocity can be derived from the attributes as x and y coordinates take into account the history of these two attributes. The modulator of the raw data also contains a selection of the relevant data that is required to compute aggregated system-wide parameters. The processed data is passed to the next components, the data analyser and predictor.

4.4.7.4 The Predictor:

The predictor is a tool or process that predicts the outcomes based on mathematics and knowledge to obtain and predict the most possible system states. It processes the data coming from the modulator and the results from the data analyser with the objective of giving a prediction of the system's future state. The predictor can use its own methods or some derived from the data analyser combined with prediction methods taken, for example, from technical analysis. Prediction includes an analysis of the system's history and QoS. For this purpose the predictor is prepared with a memory to store a given time window.

4.4.7.5 The Data analyser:

The data analyser is the process of evaluating data using analytical and logical reasoning to explore each part of the data provided based on system policy. It is gathered, reviewed and then analysed to form, as is required. The data analyser produces sophisticated solutions to analysed data. It uses and applies a set of detectors to the processed data vector. These detectors can be mathematical and statistical values to be evaluated. At the end of this step, a system-wide description of the current state is provided.

4.4.7.6 The Aggregator:

The aggregator receives data from the data analyser and the predictor. In addition, possibly some raw data emanating from the modulation is handed on with QoS to the aggregator. The aggregator then sends the outcome to the processor and evaluator model to be assessed. The evaluator has a memory to store current values together with their history and is subsequently stored forming a set of data vectors. These vectors are needed to perform filtering, such as, smoothing of the results to remove the effects of noise. Further, it delivers and sends a set of filtered current and previous values to the evaluator model via the processor.

4.4.7.7 The Processor

The processor is the last component to obtain a sequence of aggregations and linked procedures, which at every stage converts inputs into outputs. It is an organiser component and is located between the aggregator and the evaluator models. Moreover, it receives the results which contain aggregated information to be processed. It should then transfer information to the evaluator. It is used to perform a final check on the aggregated results to decide whether these results are worthwhile sending to the evaluator model or whether to disregard them.

The major aim of the e-observer is to perform an aggregation of all the available information and data about the SOM in the form of indicators to give a global description of the state and the dynamics of the underlying system, hence, the observation framework guides the e-observer.

4.4.8 The Evaluator Model

As shown in the figure 4.7, the second component in the observation model is the evaluator. The purpose of the evaluator model is to delineate the risk analysis to be carried out by detection tools. The intention is to carry out an evaluation to identify risks and problems. Furthermore, the evaluator is a model that receives results from the e-observer model to evaluate and process them in order to decide whether they adhere to the observation policy.

The figure below shows the evaluator model, which has three interfaces to organise its task and are briefly described as follows (Jürgen Branke, Moez et al. 2006): the aggregated data is obtained from the e-observer via the processor. The objectives are imposed on the evaluator by the observation system using the second interface. This is used for the evaluation routine of further actions. The evaluator contains all the information needed for the interaction and reconfiguration of the SOM. Every evaluation system provides a number of different parameters and interfaces for manipulation. During the last decision, called the action selector, the mechanisms will respond to the enforcer model via the observation model to take best action on the SOM. Therefore, the evaluator selects the best adequate actions to optimise the system's behaviour with respect to certain global objectives. The main criteria in the evaluator model are based on the behaviour and history of observed services.

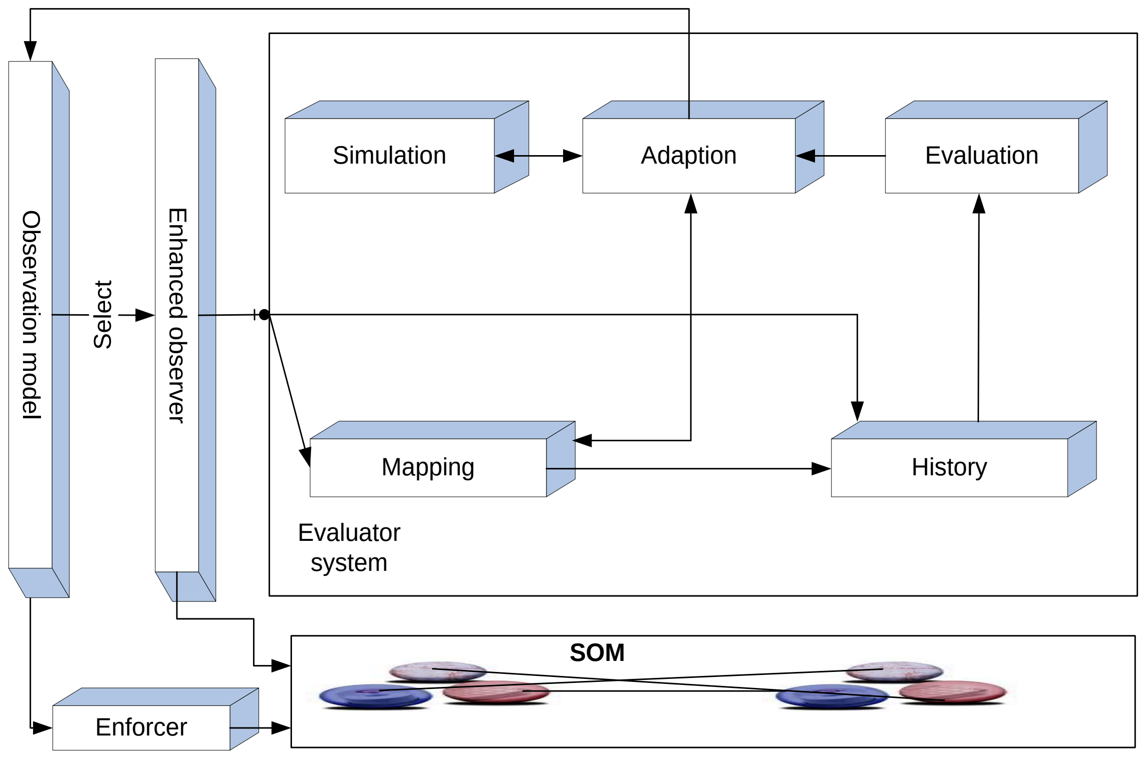


Figure 4.7. Evaluator Framework

We briefly described the main component, which is given the best possible evaluation in our approach by addressing the criteria of selected services to obtain a highly keyed solution and which then sends it to the enforcer model for detection.

4.4.9 The Enforcer Model

The third component in the observation model is the enforcer model. The enforcer executes unpleasant tasks for a superior. It enforces threats that it will not co-operate with the observation policy. It can excel at detecting adversary attempts to violate security on a host. Furthermore, it receives a final evaluated action and aggregation via the processor to perform the best trigger to the services that are determined and controlled to influence in accordance with the observation policy. In technical terms, it is an online tool that observes and identifies security deviation in the environment and then tackles and passes these on in the SOM. In addition, it uses the defined evaluation policy as input, creates a separate process as a thread for each service and runs parallel activities to detect security violations (K. Aldrawiesh, Platt et al. 2011).

The major theme behind the enforcer system is that the proactive detection provides the additional strength, which then informs the model to detect effectively any violations. The Enforcer model should be able to detect any deviation of the system. It simply works in an anti-malicious capacity to deal with intrusions and for enforcing the services or objects when required.

The popularity of the enforcer's technique has rapidly increased. Several researchers and designers developed the enforcer technique that is currently in use. Our enforcer model has some modules that have been joined to control and manage its behaviour as well as achieving an acceptable System Observer Model (SOM). These modules are commonly used to manage access control services. Moreover, the enforcer technique has been adopted by many organisations and enterprises for a variety of purposes, such as, military, business and financial. This technique is a process and a method that should detect any intrusions by applying mechanisms to control any violations.

4.4.9.1 Enforcer Architecture

This research proposes the requirements for addressing and enforcing access control policies to satisfy and reduce any violation at SOM as follows (K. Aldrawiesh, Siewe et al. 2011):

- Components: Our model consists of many components, such as, PAP, PDP, PEP and policy repository (PR). They will be described during this chapter.
- Performance: Our model makes a low time response to accomplish and support real time applications.
- Platform: Our model offers the distribution regardless of different modules on different machines and with different operating systems.
- Dynamicity: Our model can be easily extended by adding components e.g. PDP and PEP, hence additional levels of management and control should increase safety.

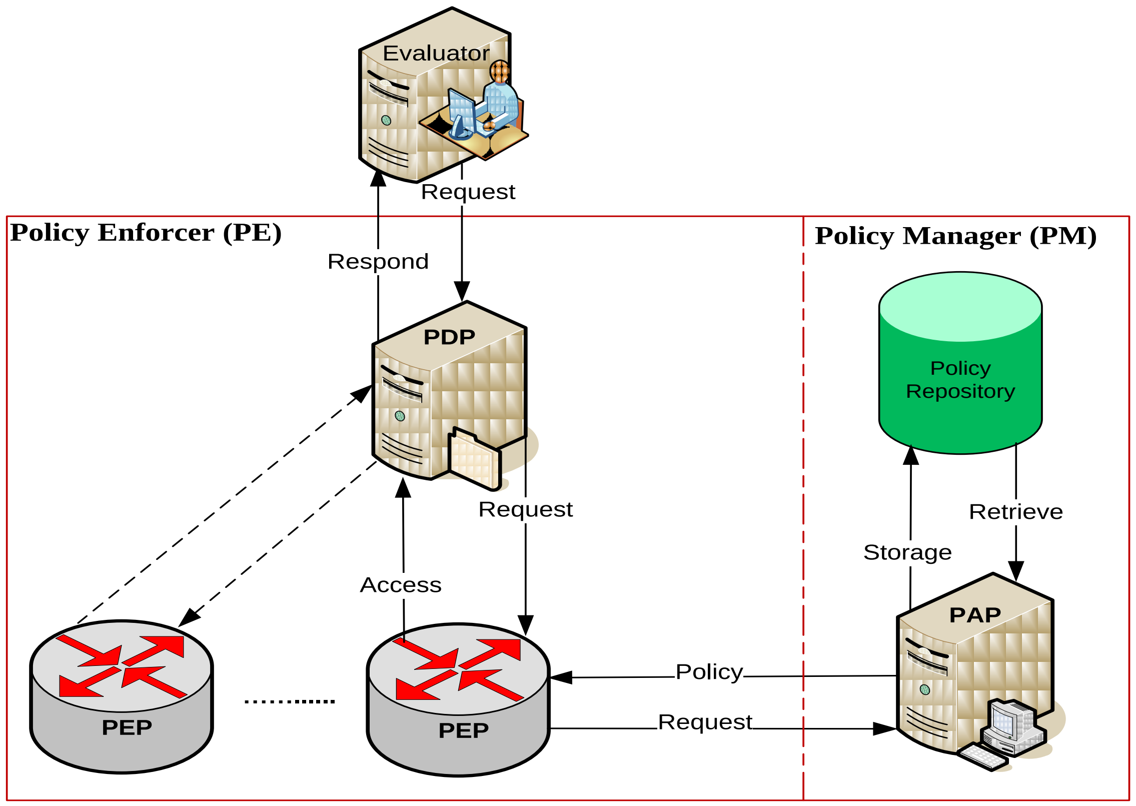


Figure 4.8 Enforcer architecture Model

As shown above in figure 4.8, the architecture is mainly contained in two parts: the policy manager (PM) and the policy enforcer (PE). This architecture totally wraps up the computation of access control services. PM is a logical unit or entity that oversees and manages policy implementation. Each PM has its own management scope. The PM is a user interface that works as offline and adjusts a list of jobs e.g. policy creation, modification, activation, synchronisation, validation and termination. The PM also maintains and updates a policy repository. The PM offers centralised administration and management of entitlement policies and delegation and integration; and uses and runs various policy validation techniques to confirm each policy before storing them in the policy-engine. The validation techniques include a syntax check, a condition check, a dictionary check and a duplication check. The PM groups policies resource wise within a domain and made ready to be input to the policy enforcer.

PE works as online tool that detects violation activities in our model and their associated components.

This architecture has set the progress of access permissions to individual Web services, returning for each request, and the decision of whether the access should be accepted or denied. In addition, it has been designed and proposed to impose the system against intruders.

As overalls of this architecture, it has various components i.e. PAP, PDP, PEP and PR. The PAP, PDP and PEP are implemented as Web services. These access control services are described in the next few paragraphs.

PAP is an entity that issues authorisation policies. PAP controls and offers centralised administration, management and the monitoring of entitlement policies, their delegation and integration. PAP creates and manages security policies and stores these policies in a policy repository (Verma 2000).

PDP is an entity that evaluates an access request against one or more policies to make an access decision. PDP assesses the prerequisite conditions for enforcing the policies with respect to their conditions. PDP is the policy management module of the router that is responsible for enforcing the deployed policies. PDP is logical unit or place on a server that enforces policies for admission control and policy decisions in response to a request to access a resource on a computer (Verma 2000).

PEP is an entity that enforces access control for one or more resources. PEP is a network device by which policy decisions are enforced. PEP is a component or unit of policy-based management. When a user tries to access a file or other resource on a computer network or server that uses policy-based access management, the PEP will explain the user's attributes to other entities on the system, then the PEP will give the PDP the decision of authorised or unauthorised user access based on the description of the user's attributes. Applicable policies are stored on the system and are evaluated and considered by the PDP. The PDP makes its decision and returns it. The PEP will inform the requester whether the requester is authorised/unauthorised to access the requested resource (Verma 2000).

PR is the main database for the policies that are managed and controlled by PAP. PR offers and provides a persistent policy store for policies that are known by the PAP. These policies may or may not be currently "active" with regard to the policy decision process. For Policy

Managers with distributed components, the PR also provides secure access. The policy repository is a set of directories that stores policies and distributes them to policy engines (Verma 2000).

The policy engine operates as a PDP as it is responsible for making decisions about the deployment of policies on the router device and is a repository for security policies in a directory store that is defined by the system.

4.4.9.2 The Description of enforcer architecture

As aforesaid in the figure above, the model and its components are implemented as Web services. The PDP component receives an access request and replies with a *'yes'* or *'no'* decision. The PEP component deals with the PAP that encapsulates the information needed to define the applicable policies from PR. It then examines the request against the applicable policies and returns the final decision to the PDP component. The PAP component attempts to action the policies that are applicable to a given access request and replies them to the PEP component.

However, a typical dialogue in figure 4.9 below shows the sequence of access control services that returns an access control decision as well as requesting attributes. Firstly, when a user requests a service to access a response to an access request, the PDP locally attempts to fulfil the request, which is dependent on applicable policies and attributes that are controlled by PIP. If it is unable to resolve the request, it becomes indeterminate or undefined, so it may require additional attribute assertions. In this case, it may query these attributes from one or more PIPs, which depend on PAP. The resulting data will be re-evaluated and the response returned to the requestor.

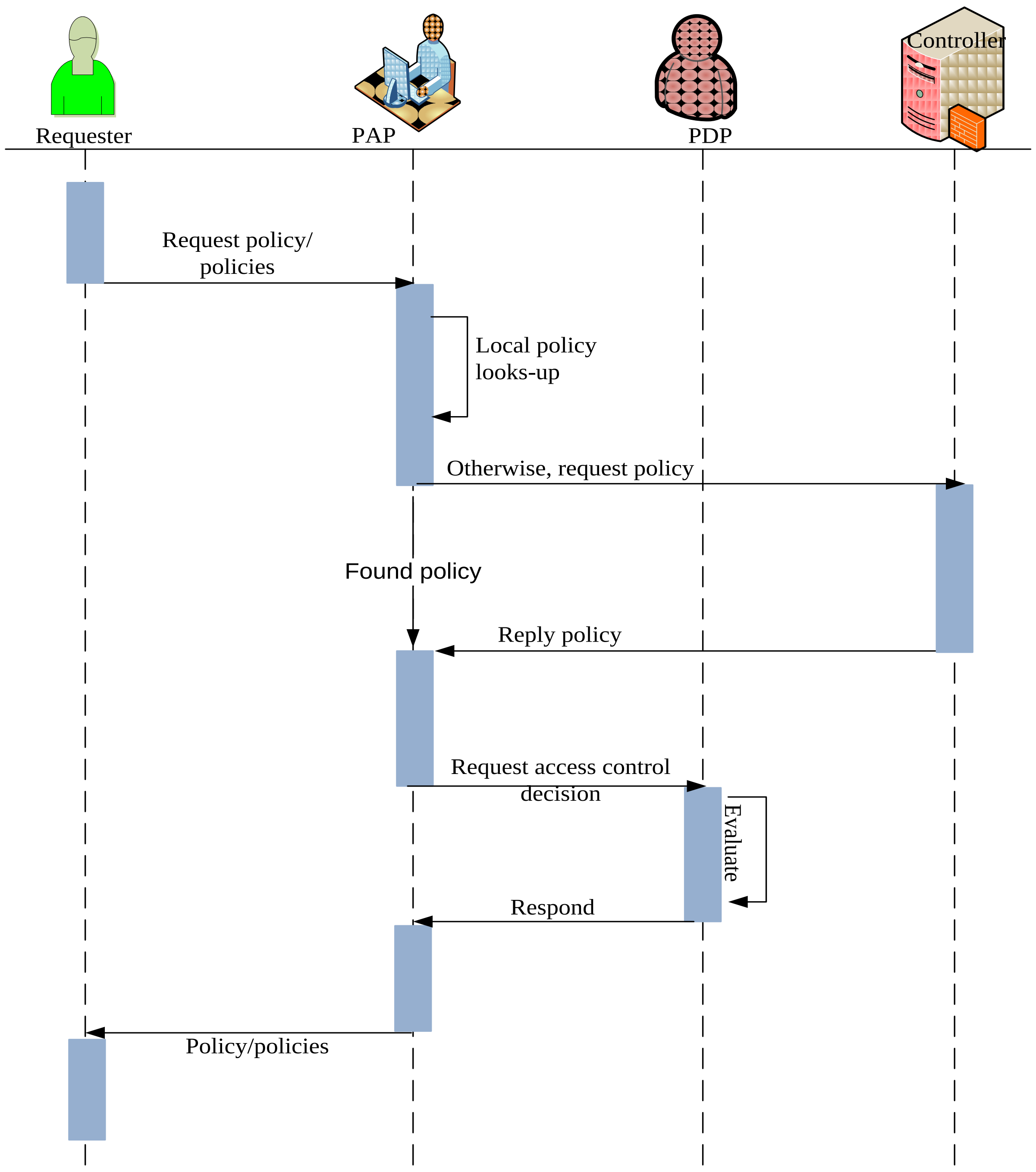


Figure: 4.9 Diagram of access control services

Moreover, figure 4.10 below, shows a typical dialogue of sequence of access control services that present a diagram to return an access control decision as well as requesting attributes. Firstly, a user requests a service to access a response to an access request. The PDP attempts to fulfil the request, which depends on the applicable policies and their attributes that are controlled by PIP. If it is unable to resolve the request, it becomes indeterminate or undefined, so it may require additional attribute assertions. In this case it may, query these attributes from one or more PIPs, which depend on PAP. The resulting data will be re-evaluated and the response returned to the requestor.

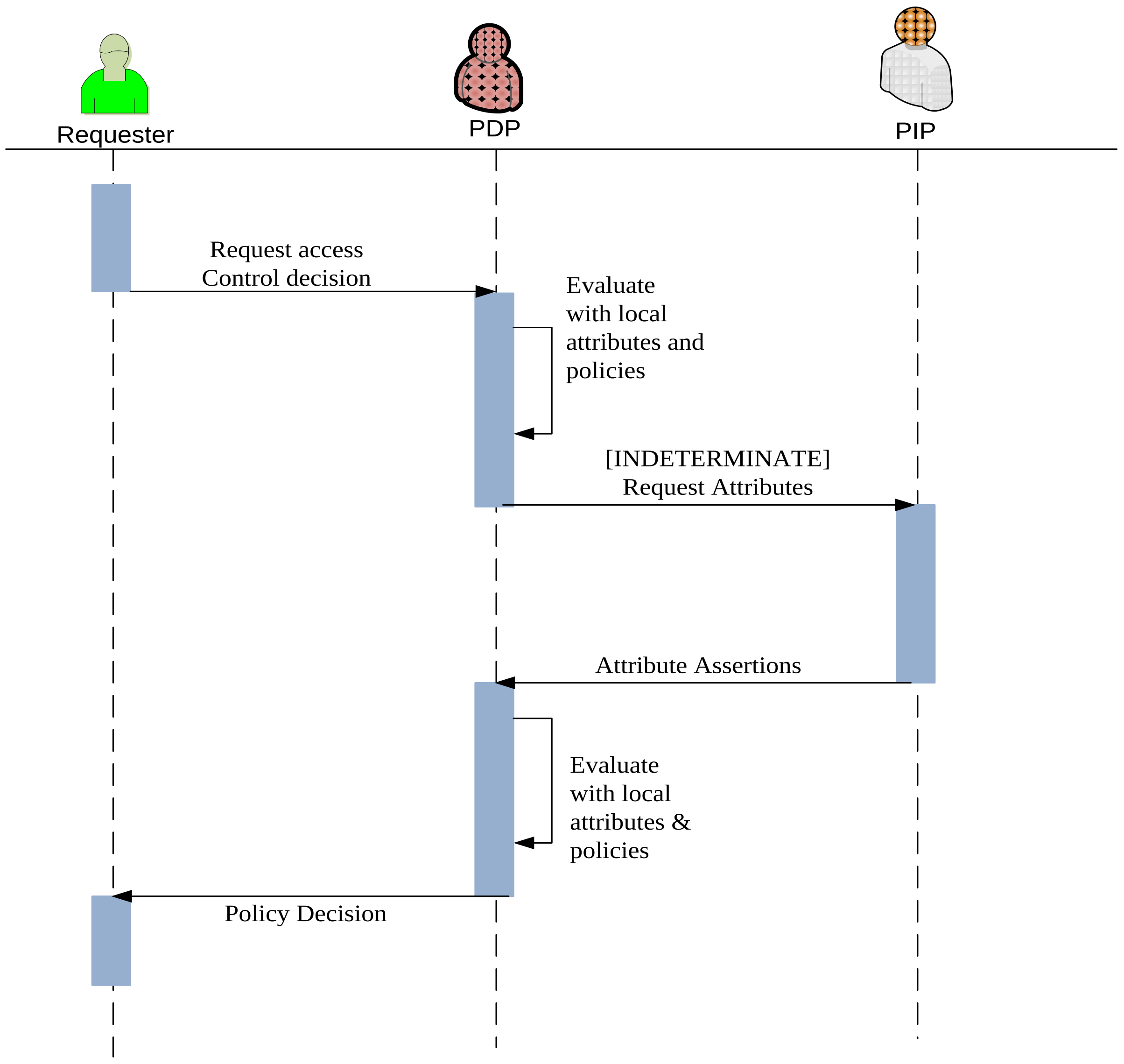


Figure: 4.10 Sequence of returning an access control decision

However, the enforcer architecture has the advantage that is dynamicity as stated in figure 4.11; it allows the model to be used in different environments that use enforceable systems. For example, if a client requests the service to access a resource; the PDP0 evaluates the request and then sends it to the PEP0, which needs to be authenticated by PAP. Next, the

PAP should generate a new PDP e.g. PDP1 and a pair of PEP1-PAPs accordingly, consequently, these components can be used and implemented as Web services.

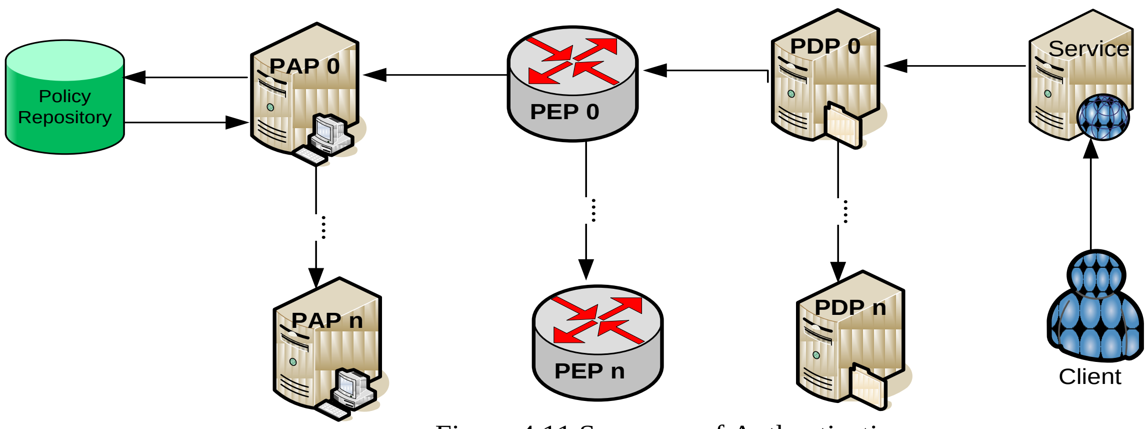


Figure 4.11 Sequence of Authentication

The major purpose here is to keep and retrieve the policies applicable to a given access request. The PAP component looks in the policy repository depending on the received parameters, such as, service name and location.

The policy manager (PM) has a user interface that generates a policy. The policy engine is a repository of security policies defined at service domains. The PM creates and produces a set of relational databases that stores policies as shown by the following table 4.1:

P. ID	P. Name	Subject	Condition	Priority	Action	Attack name

Table 4.1 Relational database stores policies

This table is set as: *Policy ID* defines a policy ID number which is unique and used to sort the policy. *Name* defines name of a policy. *Subject* identifies the range of the policy of users that are certified to initiate an action. *Condition* defines a condition to be tested for a target. *Priority* defines the priority of a policy, which ranges from zero to one (0-1). Zero is low priority and one is high priority of a policy. This priority is dynamically changeable during execution. *Action* defines the immediate action to be performed to reduce and control the risk. The *attack name* defines a suspected name that the policy can detect. As illustrated in the table, we assume an algorithm that has a collection of episodes. This algorithm can examine a policy where P is the policy as shown below.

This algorithm is used in the enforcer system at the decision stage. It performs a policy as follows: reading a policy and then obtains the value of a resource as related with a policy. Next test the condition, later if a resource gets any variation, then calls *action_policy*.

```
Begin
[Read a policy from memory]
Get←a policy (P)
[Get current values of a resource in association with policy]
   If condition ≠ satisfy
     If variation in current values of a resource
          ElseIf ∑ variation…then
Run← action_policy(P) //This will call execute action of a policy
   If policy-association exists…then
       Get←a policy from associated domain
 Endif
End
```

Algorithm 5.1 Examine the policy in Enforcer system

4.4.9.3 Policy Specification Languages

A policy is a set of declarations that is set by the owner of some computing resource. It specifies how a policy should be used. Policy specification languages (Al-Ajlan 2008) are an attempt to address and formalise the intent of the owner into a form that can be understood by machines; hence a policy may give certain permission to entities i.e. resources, programmes and users that fulfil some requirements. Each policy language must state the entities and their attributes that are to be considered, as well as the rights that can be granted. Different policy specification languages are considered and summarised for use on different networks and services with different operating systems, as well as taking into account their limitations. Therefore, a policy language is designed and proposed for easy representation of a security on a system. In chapter 5, we will elaborate on Ponder policy language.

4.5 Summary

In this chapter, we have introduced the framework of architecture that provides separate tiers for composing and coordinating services for managing services in an open environment by employing these components to obtain a meaningful advantage for enhancing the observation in the system. This model illustrates the main elements that improve the observation in the context, and can be used in different purposes that are concerned with security.

Also, in this chapter, we portrayed and described the important components and their associated issues that are used and stated in the observation model by complying optimal systemic manners to obtain a highly dependable system.

Moreover, the e-observer architectural model is a natural way of implementing and using observation. It serves the purpose to keep emergent behaviour within predefined limits for reaching the required level of security. It is a highly traditional technique that has been used by many developers for a long time and adopted by many enterprises.

The second model of the observation framework is the evaluator, which receives the outcomes of the e-observer in order to apply assessment and then writes to the third model. The evaluator should produce an evaluation report that indicates the violations which need to be detected. And then the third model which is the enforcer acts on these violations by applying detection, re-evaluation and then action. It is a highly worthable technique that works as a web service to support its detection. The advantage of the enforcer is it can easily add components e.g. PDP and PEP to manage and enhance security.

Therefore, the observation framework is responsible for actually changing the internal behaviour and structure of the processes. It possesses the major power that allows the e-observer, evaluator and enforcer to guard and control the model cycle and information by estimating the threats that could be detected by appropriate actions to adhere to the SOM.

The next chapter will consider the Ponder policy that is used in this research. This policy will be depicted in more detail in this coming chapter, and it has two main parts. The first part is access control policies, which has four parts: authorisation, information filtering, delegation and refrain. The second part is obligation policies. In addition, this chapter will contain graphical representations of Ponder policies, which also has two aspects. These are domain hierarchy policy and set operation policy. At the end of this chapter, we will explain the tool simulations problem description and AGG simulation.

CHAPTER 5

Formulate a Policy-Based Technique for the Verification of the Observation Approach

Objectives

- *To present an overview of Ponder policies and their associated parts*
- *To present a graphical representation of Ponder policy*
- *To illustrate problems by using the simulation tools (AGG)*
- *To develop compositional rules for verification of Observation system*

5.1 Introduction

The visionary promise of the usage of policy-based techniques has become an increasingly and applying solution to manage heterogeneous system networks and distributed systems. The management of large-scale integrated systems, particularly transnational system, is difficult because of the wide range of available policies with differing types of information. Due to the importance of using policy-based management, which are enhanced and developed scalability and flexibility for a system. Scalability is developed by uniformly applying the same policy to a set of devices. Flexibility is gained by separating the policy from the implementation of the managed system.

Web services are increasingly emerging as a popular standard for sharing data and functionality between loosely coupled, different platforms and heterogeneous systems. Consequently, the integration of existing Web systems' technologies permits the provision of advanced and complicated services, e.g. enabling and supporting browsers to use different sorts of resources and services at the same time in an easy procedure. However, managing and controlling vast number of services is complicated, as it needs a suitable policy to be defined in order to develop reliable and secure Web services. This chapter rests and proposes a policy verification and analysis method by graph transformations, which support an intuitive way to represent abstract policies in a simple-to-understand style. Furthermore, it performs as an informal language reference for the environment of Ponder policy language. The most important section of this chapter is the simulation of AGG tools with a set of graph rules to illustrate its ability to protect an observation system and to verify and analyse a system with the Attributed Graph Grammar (AGG) tools. This chapter shows and uses contextual graphs to define security policies that encompass actions at different stages of model systems.

5.2 The Attributed Graph Grammar (AGG)

AGG was designed and developed at the beginning of 1997. AGG is subject to ongoing research activity by the graph grammar group in Berlin. AGG (Taentzer 1997) is a development environment tool for attributed graph transformation systems supported by an algebraic approach. It aims to support and specify building prototyping applications with complex graph-structured data. AGG uses a general-purpose graph transformation engine in high level SUN JAVA applications that employs graph transformation methods. In addition, AGG language is a rule based visual language supporting an algebraic approach to graph and depict transformation. The AGG environment is designed as a set of tools to directly control, change typed and attributed graphs and to define a graph grammar.

Having and adopting an AGG graph grammar may validate using its ability analysis techniques, namely, critical pair analysis, consistency checking and termination criteria for Layered Graph Transformation Systems (LGTS).

AGG (Taentzer 1997) is a general development environment for algebraic graph transformation systems tools. AGG follows the interpretative approach. AGG has an ability that comes from a very flexible attribution concept. The relationship between Java and AGG are allowed to attribute any kind of Java objects, so graph transformations can be equipped with arbitrary computations on Java objects described by a Java expression. The environment tools of AGG include a graphical user interface comprising of a set of visual editors, an interpreter and validation tools.

Graph grammars use as specification technique for a set of particular kinds of systems, especially in situations where states exist as complex structures that can be adequately modelled as graphs and in which the behaviour involves a large amount of parallelism. They can be described as reactions to stimuli that can be observed in the state of the system. Graph grammars are a formal language suitable for a set of specifications for the type computational systems.(Taentzer 1997).This kind of formalism is particularly well suited to applications whose states have a complex topology and in which behaviour is essentially data-driven, that is, events are triggered by configurations of the state.

AGG is a technical tool that has used by a number of developers and designers for attributed graph transformation systems that support an algebraic approach to graph transformation. It is particularly suitable for rapid prototyping applications with complex graph structured data. In addition, AGG may be used as a general purpose graph transformation engine for high level SUN Java applications that employ graph transformation methods(Taentzer 1997).

The interpreter in AGG allows the stepwise transformation of graphs as well as rule applications for as long as possible. AGG supports several kinds of validations, which comprise graph-parsing, consistency checking of graphs and conflict detection in concurrent transformations by critical pair analysis of graph rules. Applications and components of AGG have graph and rule-based modelling software, validation of system properties by assigning graph transformation based semantics to some system model, graph transformation based evolution of software and the definition of visual languages based on graphs (Taentzer 1997).

With regard to AGG's abilities, it is possible to define typed attributed graph transformation with node type inheritance. This attributed type graph can be enriched by an inheritance relation between nodes. Each node type can have only one direct ancestor from which it inherits the attribute and edge types. The value of using this feature is equivalent to a number of concrete rules, which results from the substitution of the ancestor nodes by the nodes in their inheritance clan. Therefore, rules become compact and more suitable for their use in combination with object-oriented modelling(Taentzer 1997).

Thus, we will use this tool to illustrate our proposed approach by adapting the requirements to its rules in order to assess their capabilities.

5.3 Selection Study of Ponder Policy Specification Language

A policy is a set of declarations that authored by the owner of some computing resources. The declaration specifies how the policy should be used. Policies usually manage and control the behaviour of large-scale systems e.g. a Web system. Policy specification languages (Al-Ajlan 2008) are an attempt to address and formalise the intent of the owner into a form that can be understood by machines; hence a policy may give permission to certain entities i.e. resources, programmes and users that fulfil some of its requirements. Each policy language must state the entities and their attributes, and the rights that can be given. Each policy contains actions, entities, attributes and combinations but how they are represented depends upon the policy language used. A policy language, therefore, is designed and proposed for the easy representation of security on a system.

Policies can either be used in a stand-alone manner or aggregated into policy groups to act upon functions that are more elaborate. Stand-alone policies are termed policy rules. Policy groups are sets of aggregations of rules or groups, but not both. Policy groups can model intricate interactions between objects that have compound interdependencies(Syed Naqvi and Philippe Massonet 2006). Therefore, we will present a suitable policy to be used with our approach through a comparative study of the most well known policy language. Table 5.1 shows a coarse-grained of common policy specification language that has been stated and its features required for the security policy specification (Syed Naqvi and Philippe Massonet 2006; Al-Ajlan 2008).

Specification	**Ponder Policy Language**
Access control	√
Identification- Authentication	√
Confidentiality-Integrity	X
Obligation	√
Auditing	√
Delegation	√
Constraints	√
Abstraction	X
Semantics	X
Reasoning	√
Deployment	√

Table 5.1 Ponder policy specification language

5.3.1 Ponder

Ponder is popular common policy language being an objected–oriented programming language that uses a variety of access control mechanisms for firewalls, operating systems, databases and Sun Java. It supports obligation policies that are event–triggered; condition-action rules for the policy based management of networks and distributed systems. Furthermore, it can be used for security management activities.

5.3.1.1 Ponder Policies

As mentioned in table 5.1, we have chosen Ponder policy as it is suitable language for this research. In this section, we will present an extended account of this policy including its features i.e. detention, benefits and types. Ponder policies for specifying management and security policies evolved out of work on policy management at Imperial College in the UK over 16 years (E. Lupu 1998 ; Damianou, Dulay et al. 2002).

The main concepts in ponder policies are that they contain domains to group the objects to which the policies apply, roles to group policies based on a position in an organisation or enterprise, relationships to explain interactions between roles and management structures to express a configuration of roles and relationships pertaining to an organisational part (N. Damianou, N. Dulay et al. 2001). Few policy languages specify both management and security policies, such as, Ponder policy. As well as its simplicity, it uses human managers and its design model is based on domain-based policy management. Domains are categorised and are similar to directories. The advantages of using of domains are (N. Damianou, N. Dulay et al. 2000; N. Damianou, N. Dulay et al. 2001):

1. They automatically allocate data to sub-domains, which lets them handle many data sets that lead to provide scalability.
2. They allow new objects to be added /removed from the system without modifying policies.

Ponder is a set of rules that can be used to change a system's behaviour. It has a feature to present positive and negative authorisation, information filtering, obligation and delegation policies. Ponder is a set of policy specifications, which has to be compiled into a programming language for Java and C Borland (N. Damianou, N. Dulay et al. 2001; Y. Zhao and F. Parisi-Presicce 2004). As shown in the figure 5.1 below, ponder language expresses authentication to an employee.

```
Type auth+ FileAccess {subject Employee,
             Target CompanyFiles}
{
   Action read, write;
When
IsAuthorizedEmployee(TRUE);
}
```

Figure 5.1 : Authorisation actions

Before policies designed and implemented, it should be possible to analyse and verify that they behave as expected; we have to assume some definitions for the terms that are used in Ponder. These terms are subject, target, domain, action, role and relationship. They are illustrated in Table 5.2 below:

Syntax	**Expression**
Subject	Observer, evaluator, enforcer, state/service
Target	State, service, resources or objects
Domain	A group of objects, to which the policies can be applied
Action	The activities, which subject can carry out i.e. state
Role	A group of policies, which have the same subject
Relationship	A group of policies, which define the rights and duties of roles towards each other, such as, the relationship between observer and enforcer....etc
Language Keywords	Inst- subject-target-action-when-then
Choices	They are round brackets ( ) separated by
Optional elements	They are square brackets [ ]

Repetition	Braces { }
Constraints	It is optional in all types and can be specified to limit the applicability of policies based on time or values of the attributes of the objects to which the policy refers
Elements	They can be specified in any order. The subject and target elements can be optionally. This can be used to check that the objects do support the specified operations or to locate the interface specification
Policy Name	It can be specified as a path, hence defining the domain into which the policy must be stored
in/out keywords	They are used to specify input and output parameters of the action on which the filter is specified
Result	It is used to transform the return value of the action

Table 5.2: The Syntax and Expression of Ponder Policy

5.3.1.1.1 Types of Ponder Policy

The Ponder language is able to specify security policies that portray a wide range of access controls that implement mechanisms for firewalls and operating systems. Ponder adopts and copes the language by making it flexible and extensible for a wide range of management requirements and actions them on the current platforms of distributed systems. An object-oriented language that specifies management for distributed system patterns. Ponder language has two main types of policies (N. Damianou, N. Dulay et al. 2001; Y. Zhao and F. Parisi-Presicce 2004):

- ✓ Access control policies
- ✓ Obligation policies

5.3.1.1.1.1 Access Control Policies (ACP)

ACP is defined as the most fundamental and widely used security mechanisms. It authorises users to perform a set of actions on a set of resources within a system as well as having the ability to permit or deny the use of a particular resource by a particular entity. It can be used to manage several types of resources e.g. physical, logical and digital. Further, it is concerned with limiting the activity of permitted browsers who have accessed successfully.

All policies relate to objects by using an interface definition language in which the expression subject refers to browsers and to administrators/owners who have management responsibility. The subject accesses target objects by invoking methods visible on the target's interface. The most suitable protection for ACP in Ponder is an interface method and both subject and target objects are stored in the domain's service. These domains supply a means of grouping objects to which policies apply and they can be used to divide the objects in a huge system according to object type, authorisation and responsibility and for the convenience of human managers. The policy specifications, with millions of objects in large-scale systems, are similar to directories and are implemented using the Lightweight Directory Access Protocol LDAP service. Ponder policy offers ACP in the form of four policy sub-types, which are described as (Evan 2007; S. Parsa and M. Damanafshan 2007; V.. Hu, E. Martin et al. 2007):

➢ **Authorisation Policy**

The authorisation policy is central and the other sub-types are considered as auxiliaries. This type of policy identifies access control for security. It describes a member of the subject domain that can access the group of objects in the target domain and this leads to protect services and resources from unauthorised access. In addition, it has two types:

- A negative authorisation policy that identifies the actions that subjects are forbidden to undertake on target objects.

- A positive authorisation policy that identifies the actions that subjects are permitted to undertake on target objects.

Both these types, positive (*P*+) and negative (*P*-) authorisation policies must contain the following policy parts:

- Subject (except in roles)
- Target
- Action

The authorisation policy is presented and implemented on the target host by an access control component. The syntax of the authorisation policy structure is shown in the figure 5.2, below the expression of this structure is described in Table 5.2. Here, we have stated an example of an authorisation policy, a positive one (P+) and a negative one (P-):

```
Inst       ( auth+ | auth– )       policyName    "{"
        subject       [<type>]     domain-Scope-Expression ;
        target       [<type>]      domain-Scope-Expression ;
        action                     action-list ;
        [ when                      constraint-Expression ; ]
"}"
```

Figure 5.2 : Authorisation Policy Syntax

- **Information filtering Policy**

There is a need to filter the information policy because filtering should transform the data input/output parameters into action. For example, a location service can authorise access to detailed position information, e.g. a person in an exact area to browsers within the sub-division. Outside browsers can only determine whether someone is at work or not. Authorisation policies (positive P+) may include filters to transform input or output parameters as well as with their actions based on attributes of the target or subject on system parameters i.e. time. The process has to be completed and then a choice made on whether to permit the results to be returned to the subject or whether the results need to be transformed, therefore, filtering policies can only be used and applied to positive authorisation actions.

```
actionName    { filter }
  filter   =    [ if   condition ]   "{"   {  ( in parameterName = expression ; |
                                                 out parameterName = expression ;
                                             |
                                                 result = expression ; ) }
                                      "}"
```

Figure 5.3: Filters on Positive Authorisation Actions

As stated in figure 5.3 above, each action can be related to a number of filter expressions. An optional condition should include every filter but if the condition is assessed as being true then the transformations are executed.

➢ Delegation Policy

Delegation in general is a given authority and responsibility to another i.e. user/service/unit. It carries out specific activities and rights and it will remain accountable for the outcome of the delegated work. Delegation means to empower a subordinate to make decisions.

A delegation policy is kind of permission to a subject. It gives the subject actions to be delegated to others but a user must be strongly managed and controlled by security policies to delegate access rights to another. This kind of policy authorises subjects to award privileges or rights, which they have to grant to achieve an action on their behalf; it enables cascaded delegation of access rights and it is critical in software.

```
inst deleg+ “(”associated-auth-policy “)”  policyName    “{”
       grantee      [<type>]     domain-Scope-Expression ;
         [ subject   [<type>]     domain-Scope-Expression ; ]
         [ target    [<type>]     domain-Scope-Expression ; ]
         [ action                 action-list ; ]
         [ when            constraint-Expression ; ]
         [ valid                  constraint-Expression ; ]
 “}”
```

Figure 5.4: Syntax of Delegation Policy

In addition, a delegation policy is defined as a permission policy that classifies access rights to resources. A delegation policy does not mean using a system that has specified user rights by security administrators.

As stated in figure 5.4, according to the syntax of a delegation policy, the delegation has two categories: policy positive and negative where negative delegation policies ban delegation. The delegation policy has only one require part, that is, the grantee type. This type has three parts: subject, target and action. These parts must be sub-sets of those in the associated authorisation policy. A positive delegation policy classifies delegation constraints to limit the validity of the delegated access rights. Constraints could be time restrictions, together with duration and validity periods in order to indicate the length over which the delegation is valid before it is revoked.

➢ Refrain policy

Refrain policies indicate to a subject to refrain from doing something and are similar to negative authorisation policies but are interpreted by the subject. These kinds of policy define the actions that the subjects have to refrain from achieving even though they may actually be permitted to access that target.

```
Inst refrain testingRes {
        subject         s=/test-programmers ;
        target          /analysts + /designers;
        action          discloseTestResults() ;
        when            s.testing_sequence = "in-progress" ;
}
```

Figure 5.5: Refrain Policy Syntax

There is a similarity in the syntax of refrain and negative authorisation policies. Refrain policies are compulsory for subjects but not for access controllers. They also take action as restraints on the actions that subjects accomplish and are executed by those subjects. Furthermore, they are used for situations where negative authorisation policies are unsuitable, as the targets cannot be trusted to enforce the policies. Figure 5.5 illustrates the similarity of the syntax used in refrain and negative authorisation policies.

5.3.1.1.1.2 Obligation policies

Another type of Ponder Policy is an obligation policy, which states the activities that a subject has to do in order to a set number of target objects and they define the tasks or duties of the policy subject. In addition, these types of policies are managed and interpreted by an administrator at the subject and describe the behaviour of the managers of the system when an event happens. Obligation policies define the actions that need to be executed by policy managers when specific events happen and they support the ability to respond to changing situations.

Moreover, obligation policies are event-triggered and identify the activities that subjects, e.g. what human or automated manager components have to accomplish on objects in the target domain. Events may be internal or external timer ones that are alerted by monitoring service components. Composite events can be stated by using event composition operators. The required event specification follows the *'on'* keyword as stated at the syntax of obligation policies in figure 5. 6.

```
Inst   oblig policyName    "{"
       on                  event-specification ;
       subject   [<type>]  domain-Scope-Expression ;
       [ target  [<type>]  domain-Scope-Expression ; ]
       do                  obligation-action-list ;
       [ catch             exception-specification ; ]
       [ when              constraint-Expression ; ]
"}"
```

Figure 5.6: Obligation Policy Syntax

In obligation actions, the aim of the element is optional (it might be internal to the subject), while authorisation actions always relate to a target object. Actions have to be preceded by a prefix representing the target set when they are be invoked on a target. The catch-clause is optional and describes an exemption that is executed if the actions are unsuccessfully performed for some reason.

5.3.1.2 Graph Representation of Ponder Policy

Ponder includes tools for dynamically managing and controlling the behaviour of system components that are devoid of changing codes. This feature requires the co-operation of the components that are being governed. A system should be able to adapt continuously in order to adjust to differences in externally imposed constraints and environmental conditions. A policy-based approach requires a suitable policy representation and the design and development of a policy management framework for controlling a system and then policies will gradually become more important to the real-world implementation of Web services(Damianou, N. Dulay et al. 2001; Parisi-Presicce and Zhao 2005). In this part of the chapter, we will discuss the representation of the basic policy types of Ponder by graph rules.

- **Domain**

Domain is a collection of entities used to group the system's data. They are controlled and managed by all the policies. It is arranged in a hierarchical manner, represented by a graph, with nodes for every domain and edges representing the sub-domain relationship of an object to a domain. As shown in the figure 5.7, our graph representation has six domains (D1, D2 D3, D4 D5 and D6):

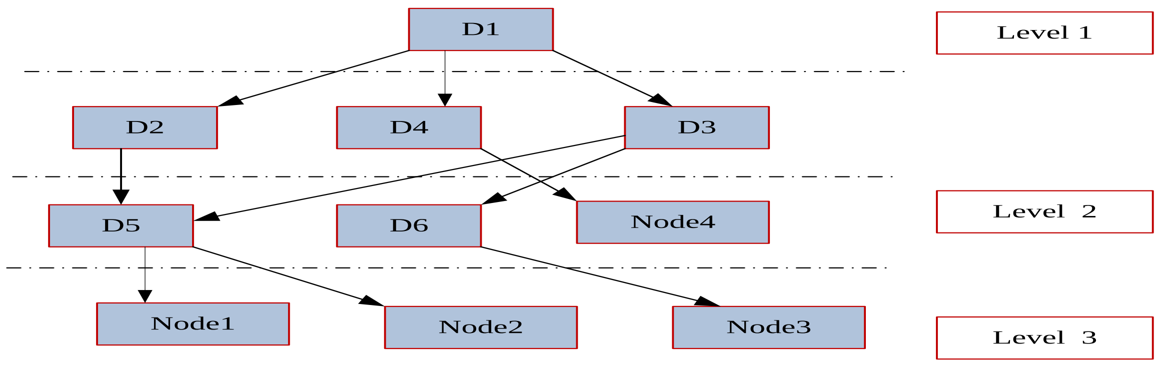


Figure 5.7: The Search paths in Domains

The domain paths as declared displayed and processed in table 5.3 For instance, if our goal is node 3, the search should take the first or third path in table 5.3 D1/D2/D5/node1 or D1/D3/D5/node 1.

No	Path	No	Path
1	D1/D2/D5/node1	4	D1/D3/D5/node2
2	D1/D2/D5/node2	5	D1/D3/D6/node1
3	D1/D3/D5/node1	6	D1/D4/node1

Table 5.3, the domain paths as in figure 5.7

- **Operation**

As shown in figure 5.8 below, Ponder uses many symbol/operations i.e. union (**U**), difference (–) and intersection (∩), to represent domain ScopeExpressions in a policy. This figure presents the operations of Ponder policies for the graph representation of domains in figure 5.7 above.

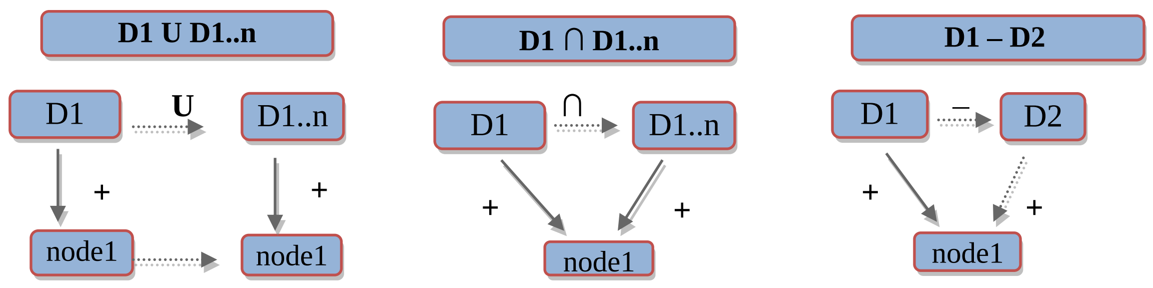


Figure 5.8: The Operation in Ponder Policy

As they appeared in the figure 5.8,

1. The left stage states the union (U) operation (D1 U D1...n), with one graph for every component. The adding symbol '+' means that there exists at least one edge in the path from a domain to a node. While a node satisfies at least one of the graphs, it satisfies the union.
2. The middle stage shows the intersection (∩) operation (D1 ∩ D1…n). It satisfies the whole graph if a node satisfies the intersection operation.
3. The right stage presents the difference (–) operation (D1 – D2). The dashed edge symbolises the negative constraint and the solid edge symbolises the positive constraint, such as, the absence of a path from D2 to node1.

5.3.1.3 The Specification and Verification of Simulation Tools

Simulation tools are a method of computational model or computer programme that tries to imitate an abstract model of a particular system. Simulations tools have become a powerful technique in most fields of modelling and it has enabled several systems to achieve insights into their operations as well as observing their behaviour. Further, simulations are useful for modelling and analysing the factual performance of evolving production systems(Azadeh, Anvari et al. 2007). A simulation environment enables designers and developers of established systems to predict behaviour and apply the tools needed to manage disturbances to an acceptable degree. Thus, simulation tools should intelligently direct a production system towards a smoother and more efficient performance. In addition, simulation tools should enable the automatic prediction of the behaviour of most systems as well as increasing production situations.

This section describes a simulation tool, demonstrated by the AGG tool. This tool will portray a simplified observation problem with interaction between services/states by using a proposed approach. We will, therefore, explain the problem and then use the AGG tool to create and propose a set of rules. Figures 5.10 and 5.11 show the relationship between the components that are used in the observation model as well as stating the scenario for most surveillance platforms.

The syntax is similar to that used for Ponder policies (authorisation policy type or refrain policy type) as in Figures 5.2 and 5.5 to design and declare our policy as shown in Figure 5.9. This because these types of policies specify the actions that subjects must refrain from performing on target objects. For instance, if states enter the System Observer Model (SOM), the e-observer will observe these states at every rotation at SOM. Thus, we can use our policy to specify this activity, as in Figure 5.9. The syntax of the observation model contains some definitions for the terms that are used in the policy. These terms are *SUBJECT*, *TARGET*, *IF*, *WHEN* and *ACTION*. All these terms were explained above in Table 5.2.

```
POLICY        policy_name        {
       SUBJECT      object ;
      TARGET       object ;

      WHEN        trigger ;
      IF          condition ;

      ACTION       action ;
}
```

Figure 5.9: The Syntax of Observation Model

5.3.1.4 The Explanation of Problem

An understanding of the relationships between the nodes is important for building the scenario of our observation model. Therefore, before starting to analyse the problem, the model should know the position of every node as shown in Figure 5.10. The e-observer is the main component in the architecture. It can invoke any target, such as, update or delete. It has the ability to manage the sequence of the data flow that is coming through SOM and accessing the monitor component (Described in chapter 4) which is a part of the e-observer. The data then experiences a long trip via the activities of several components. Figure 5.10 simply presents the relationship between the components and state in the Observation Model.

Figures 5.10 and 5.11 below both show the three main components of the observation model: e-observer, evaluator and enforcer. The state, however, is the service that comes via SOM to be observed.

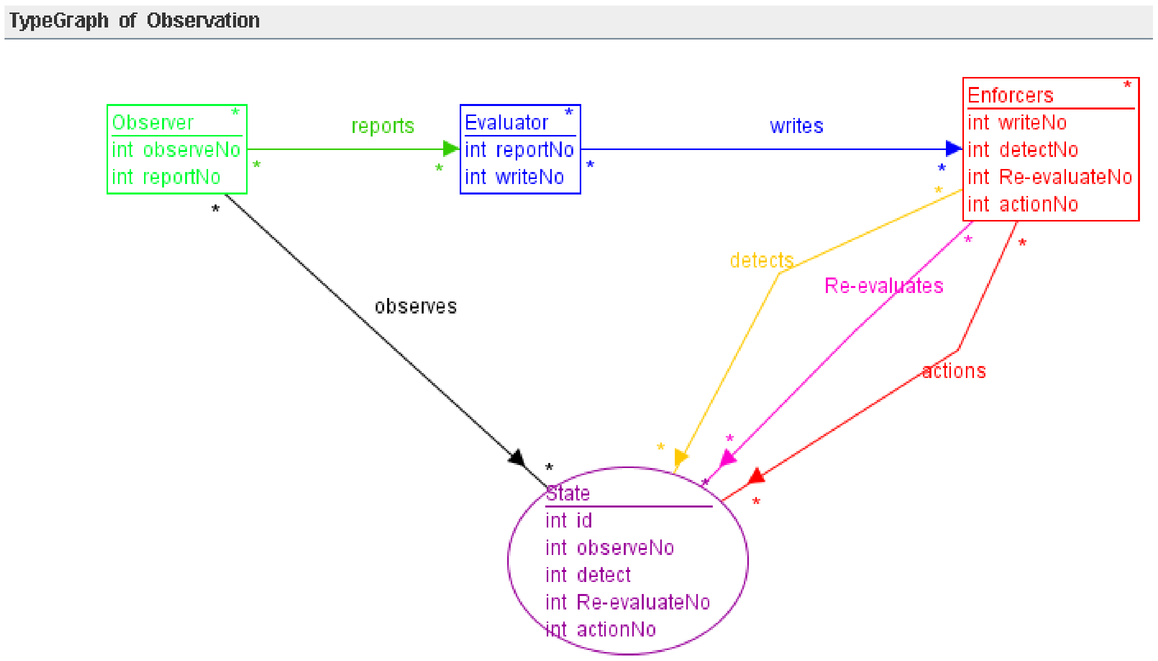


Figure 5.10: An AGG diagram between e-observer, evaluator, enforcer and state of the Observation Model

5.3.1.5 AGG Simulation

As abovementioned, the AGG is a technical tool used to detect and control conflicts either statically or dynamically. AGG defines three types of conflicts (Azadeh, Anvari et al. 2007) which it can find during static checking:

1. One rule application deletes a sub-graph needed for the match of another rule application.
2. One rule application generates a sub-graph prohibited by a negative application condition of another rule application.
3. One rule application changes the attributes needed for the match of another rule application.

Figure 5.11 shows that the graph of the observation model that has been designed by using AGG tools. This graph shows and describes the scenario of the problem, which is with some type nodes: E-Observer, Evaluator, Enforcer and State. The most important nodes in our type graph are e-observer. We shall describe this type of node with their rules and policies below.

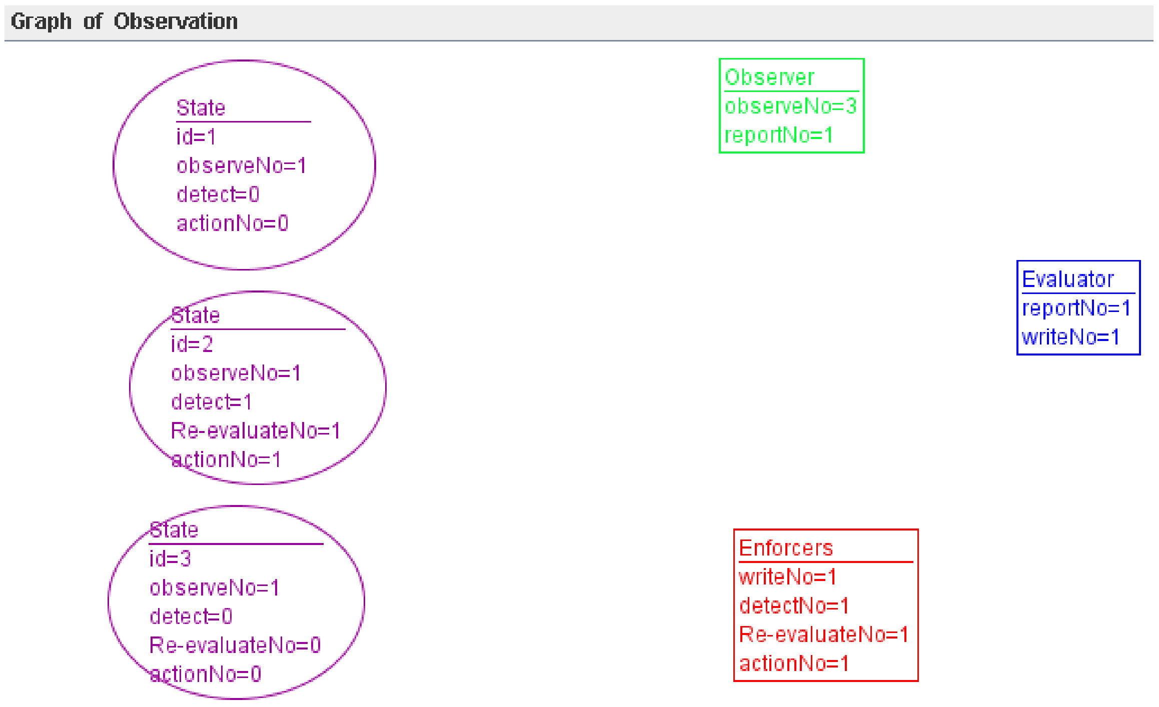


Figure 5.11: The Graph of Observation Model

5.3.1.6 The Rules and their Policies of the Observation System

In this part, we established some rules regarding the scenario in the type graph of observation as showed above in Figure 5.11. A policy was built for each rule, which depended on the policy syntax, as in Figure 5.9. The tool that was used to set up these rules was an AGG tool as described in the above section. A list of abbreviations (*o, e, en and s*) in our policy indicates e-observer, evaluator, enforcer and state respectively. The left cell in each rule is the negative application condition, the middle cell is the left side and the right cell is the right side.

Here, we will state our rules as the following; the first rule gives the e-observer node the right to observe the state's node. The second rule allows the e-observer node to report to the evaluator node with the condition: if observer = observes(s) = true; then the policy enables the observer to report to the evaluator. The third rule enables the evaluator node to write a note with a condition: if evaluator = o.report(e) = true; then the policy enables the evaluator to write a note. The fourth rule enables the enforcer node to do a detect for the state node with the condition: if enforcer = en.write(en) = true; then the policy enables the enforcer to do a detection. The fifth rule enables the enforcer node to do a re-evaluate for the state node with the condition: if enforcer = en.write(en) = true; then the policy enables the enforcer to do a re-evaluation. The sixth rule enables the enforcer node to do an action for the state node with the condition: if enforcer = en.write(en) = true; then the policy enables the enforcer to do an action. These rules and their policies are as follows:

State Observation Rule

This rule permits the e-observer node to observe the state node as in Figure 5.12 below. There is no condition for this rule. The middle cell is the left side and the right cell is the right side.

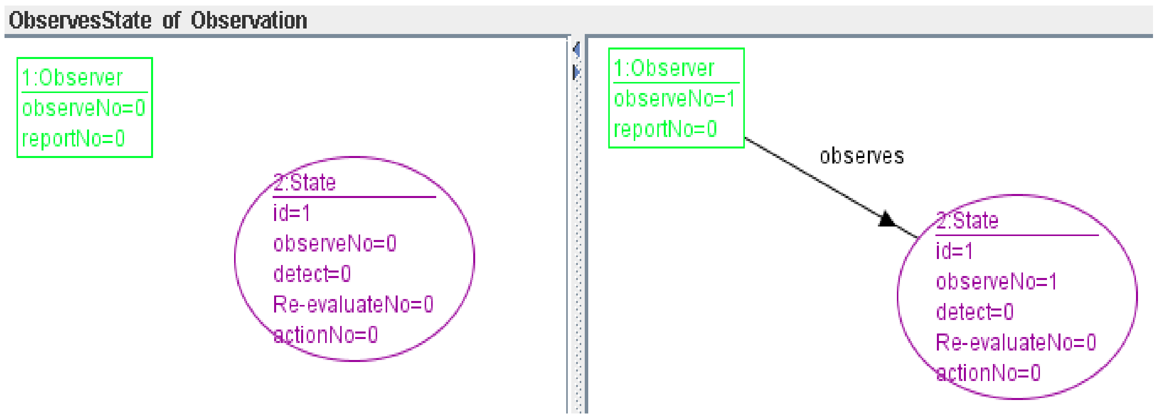


Figure 5.12: Rule 1-The Observation of State

Rule 1: *State Observation*

Description: *This rule says that an e-observer observes a state from many states as in Figure 5.10*

Parameters*: Observer (observeNo, reportNo) and State (id, observeNo)*

The policy of rule 1 is described in Figure 5.13. This policy allows an observer to observe a state node. This policy does not have a condition to observe the state and the observer will observe all the states.

```
POLICY      observation   {
        SUBJECT     o= observer;
        TARGET      s= state;
        ACTION      get state observation;
}
```

Figure 5.13: The Policy of State Observation

E-Observer Report Rule

Figure 5.14 below describes the second rule, which permits the observer node to write a report that describes the condition of the state. The condition of this rule is o.observe, so only the observer node must observe the state node to satisfy this rule.

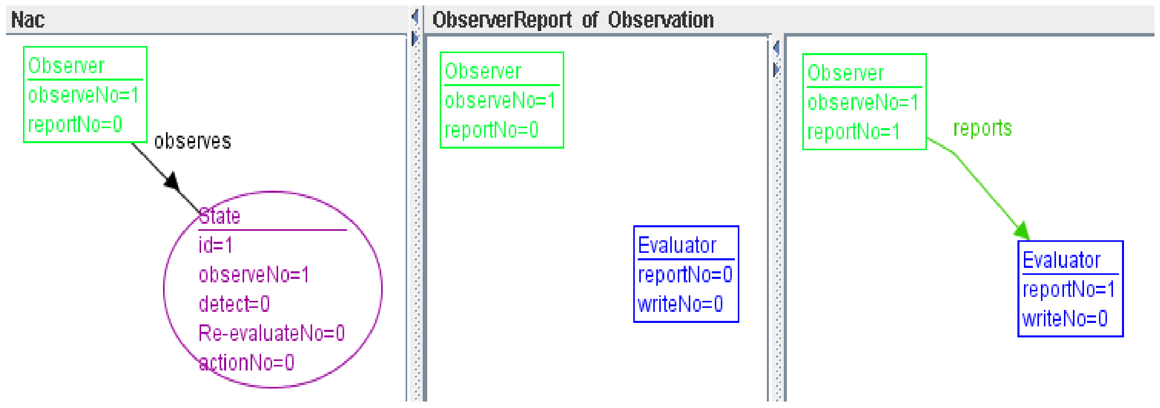


Figure 5.14: Rule 2 - the Report of Observer

Rule 2: *Observer Report*

Description: *This rule enables the observer to report to the evaluator for all states.*

Parameters: *Observer (observeNo, reportNo) and Evaluator (reportNo, writeNo).*

The policy of rule 2 is described in Figure 5.15. This policy allows an observer to report to the evaluator only if the observer meets the condition: if observer = observes(s) = true; then the policy enables the observer to report to the evaluator.

```
POLICY    Observer_Report   {
        SUBJECT    o= observer;
        TARGET     e= evaluator;
        WHEN       o.reports(e);
        IF         o.observes(s)
        ACTION   get observer report;
 }
```

Figure 5.15: The Policy of Observer Report

Evaluator Notes Rule

The third rule shows how an evaluator can write a note to the enforcer depends on the report of the e-observer as shown in Figure 5.16. The condition of this rule is o.report, so only the e-observer node must observe state node to satisfy this rule. If e-observer satisfies this condition, the rule will allow the binding service.

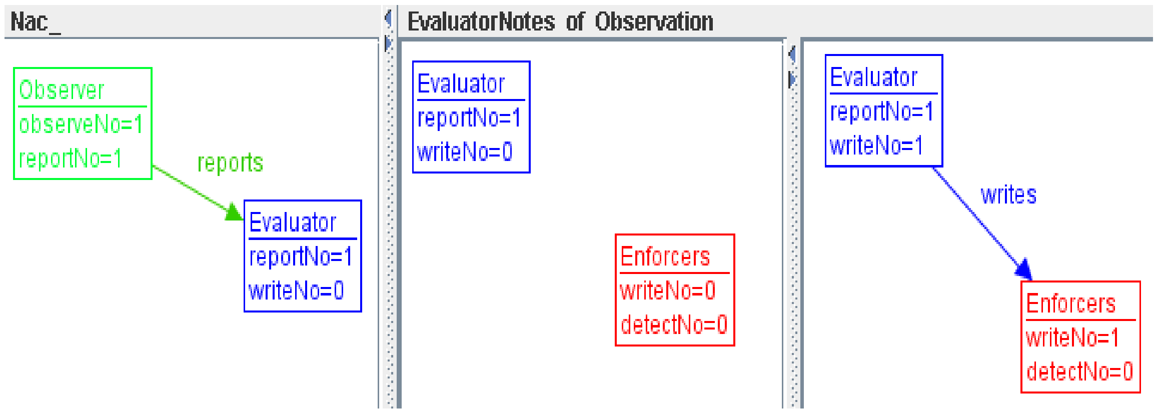


Figure 5.16: Rule 3 - Evaluator Notes Rule

Rule 3: *Evaluator Notes*

Description: *This rule enables the evaluator to write a note to enforcer and this depends on the previous action of the observer, which is o.reports (e).*

Parameters: *Observer (observeNo, reportNo), evaluator (reportNo, writeNo) and enforcer (writeNo, actionNo).*

The policy of the evaluator notes rule is described in Figure 5.17. This policy allows an evaluator to write a note to the enforcer node only if the evaluator meets the condition: if evaluator = e.write(en) = true; then the policy enables the evaluator to write a note.

```
POLICY        Evaluator Notes          {
          SUBJECT     e= evaluator;
          TARGET      en= enforcer;
          WHEN        e.write(en);
          IF          o.reports(e);
          ACTION      get evaluator notes;
}
```

Figure 5.17: The Policy of Evaluator Notes.

Enforcer Detection Rule

The fourth rule shows how the enforcer detects the state as shown in Figure 5.18 below. The condition of this rule is that the enforcer must e.write(en). If the enforcer satisfies this condition, the rule will allow the enforcer to the next rule, which is to re-evaluate the state.

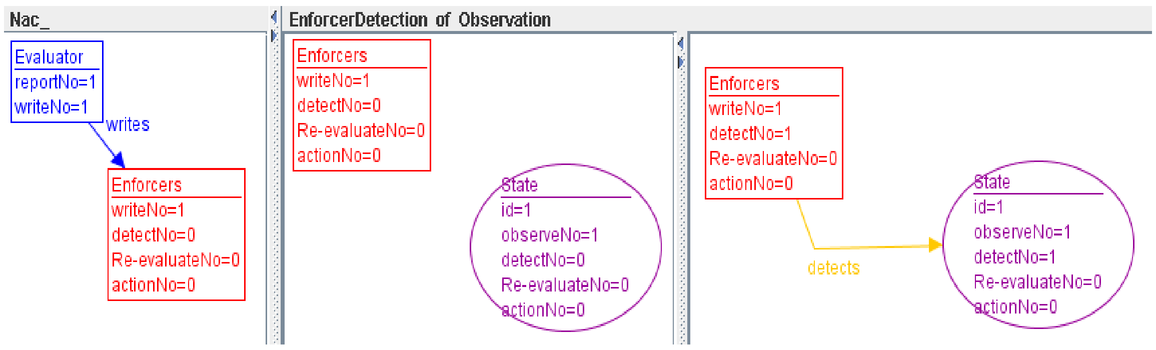


Figure 5.18: Rule 4 - Enforcer Detection Rule

Rule 4: *Detect the state by the Enforcer.*

Description: *This rule enables the enforcer to detect the state if the enforcer already has a note from evaluator that the state has been observed.*

Parameters: *enforcer (writeNo, detectNo, Re-evaluateNo, actionNo), evaluator (reportNo, writeNo) and state (id, observerNo, detectNo, Re-evaluateNo, actionNo)*

The policy of rule 4 is described in Figure 5.19 below. This policy allows an enforcer to detect a state node only if the enforcer meets the condition: if enforcer = en.write(en) = true; then the policy enables the enforcer to do detection.

```
POLICY      Enforcer Detection of State     {
        SUBJECT    en= enforcer;
        TARGET     s= state;

        WHEN       e.write(en);
        IF         en.detect(e);

        ACTION     get  detect state;
}
```

Figure 5.19: The Policy of Enforcer Detection of State.

Enforcer Re-evaluation Rule

The fifth rule shows how the enforcer re-evaluates the state as shown in Figure 5.20 below. The condition of this rule is that the enforcer must e.write(en). If the enforcer satisfies this condition, the rule will allow the enforcer to the next rule, which is an action state.

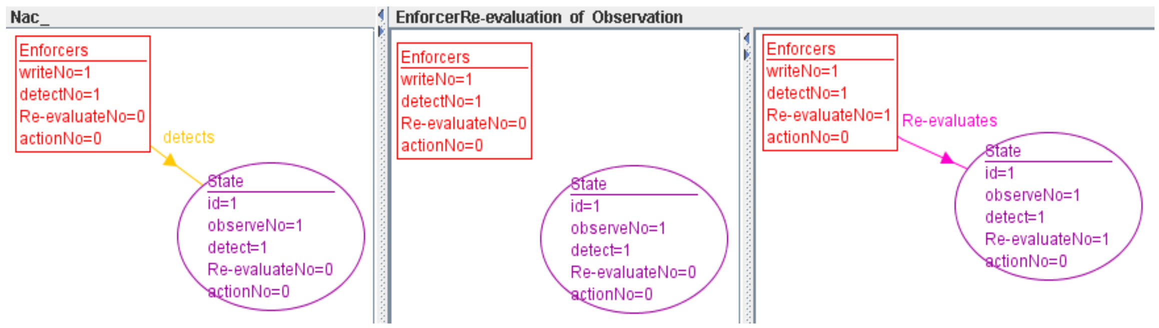


Figure 5.20: Rule 5 - Enforcer Re-evaluation Rule

Rule 5: *Re-evaluate the state by enforcer.*

Description: *This rule enables enforcer to re-evaluate the state if the enforcer already has detected the state after the note from the evaluator states that the state has observed.*

Parameters: *enforcer (writeNo, detectNo, Re-evaluateNo, actionNo), evaluator (reportNo, writeNo) and state (id, observerNo, detectNo, Re-evaluateNo, actionNo)*

The policy of rule 5 is described in Figure 5.21. This policy allows an enforcer to detect a state node only if the enforcer meets the condition: if enforcer = en.detect(s) = true; then the policy enables the enforcer to do detection.

```
POLICY      Enforcer Re-evaluation of State      {
        SUBJECT     en= enforcer;
        TARGET      s= state;

        WHEN        en.re-evaluate(s);
        IF          en.detect(s);

        ACTION      get re-evaluate state;
}
```

Figure 5.21: The Policy of Enforcer Re-evaluation of State

Observation Action Rule

The sixth rule shows how the last step of our scenario, which is an enforcer, can make an action for a state that is observed by the e-observer, as shown in Figure 5.22 below, the condition for this rule is that the enforcer must en.re-evaluate (s). If the enforcer satisfies this condition, the rule will allow the observation action.

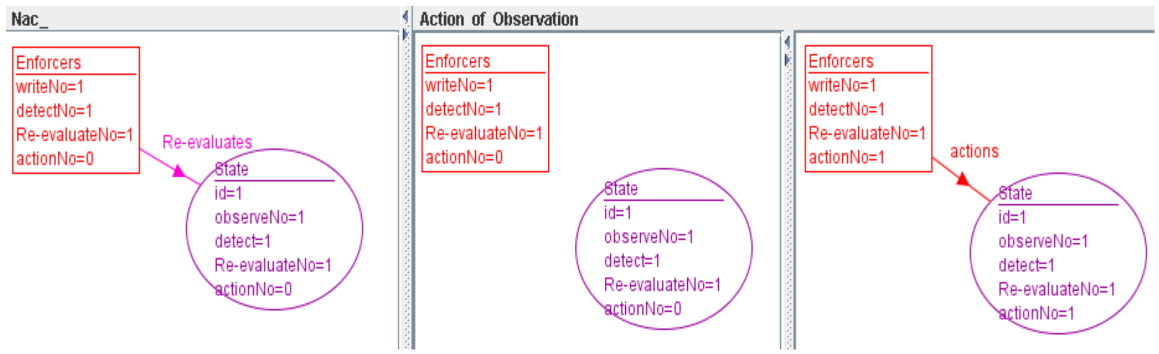


Figure 5.22: Rule 6 - Observation Action Rule

Rule 6: *Action of Observation*

Description: *This rule enables the enforcer to do an action if the enforcer already has note from evaluator that the state has observed.*

Parameters: *enforcer (writeNo, actionNo), evaluator (reportNo, writeNo) and state (id, observerNo)*

The policy of rule 6 is described in Figure 5.23 below. This policy allows an enforcer to do an action on a state node only if the enforcer meets the condition: if enforcer = en.re-evaluate(s) = true; then the policy enables the enforcer to do an action.

```
POLICY        Action of Observation          {
      SUBJECT       en= enforcer;
      TARGET        s= state;
      WHEN          en.action(s);
      IF            en. re-evaluate (s);
      ACTION        get action;
}
```

Figure 5.23: The Policy of Evaluator Notes

5.3.1.7 The Result of Execution Rules

By using AGG tools, we can execute our rules with their conditions and the results of this execution are shown in Figure 5.24 below. This figure highlights and describes how to overcome the infected states after a collection of analysis and operations. It shows the components of the observation model when makes the compilation by using AGG tools. In addition, the outcomes here is that the e-observer observes the states and then reports to the evaluator in order to evaluate these interactions and next, writes to the enforcer for detection and enforcement. The enforcer has three rules; detect, re-evaluate and afterwards make action on deviation.

Moreover, it is possible to verify formally the security policies that are captured in our observation diagram by proposed approach. This is done and validated by AGG tools. AGG possess an excellent ability to transfer the graph grammar so conveying the structure and operation of many types of systems. AGG formulates and then examines our proposed rules with their attributes on associated components to obtain the optimal outcomes for the verification of observation diagrams on a number of entities in one test. This enables to discover whether the security policies are consistent with the accompanying Ponder rules and viewing what actions should entities be subjected.

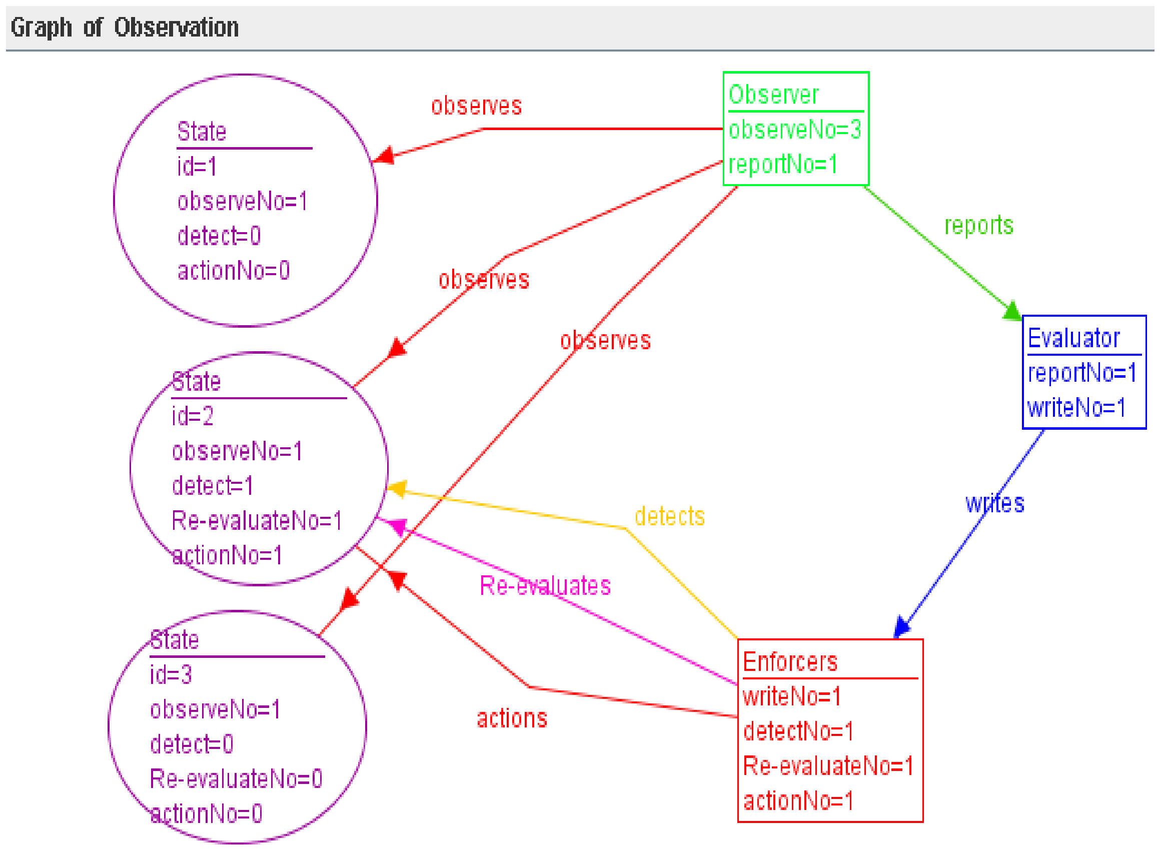


Figure 5.24: The Result of Execution Type Graph on Observation Model

5.4 Summary

Despite rapid advance in enforcing violation activities and the efforts of developers and researchers, the demand for new and updated requirements continues to outstrip available technological solutions.

This chapter presented a comparative study of a selection of well-known policy specification languages Due to this comparison study of the most well-known policy languages, which has chosen Ponder as a suitable policy specification language to be used it in this research. Ponder is a language for specifying policies for the management and security of distributed systems. Ponder includes most standard features, such as, authorisation, filter, refrain and delegation policies for specifying access control and obligation policies to specify management actions. Ponder thus provides a uniform means of specifying policy that relates to a wide range of management applications, networks, storage, systems applications and service management. In addition, this research has presented a proposed technique that addresses and supports the analysis of policies. This technique aims to allow feasible validation of policies in their development life cycle.

Moreover, this chapter has proposed a Ponder policy verification and analysis method by graph transformations. It acts as an informal language reference for the environment of the Ponder policy language. We have presented an overview of Ponder including it language sub-types and a brief comparative study with several other policy languages. We have discussed and showed in detail the representation of Ponder policies by graph rules and has used the simulation (via AGG tools) with a set of these rules. Subsequently a policy example was used to show how to verify and analyse our proposed approach by using AGG tools. These tools have the capability and efficiency to represent the rules of graph transformation systems supported by an algebraic approach.

This chapter demonstrated that it is possible to model partially the observation diagram with its accompanying Ponder rules. This means that it is feasible to verify formally the security of graphical tools.

In the next chapter, we will describe in depth the development of enforcement technique for security violations, which illustrate how enforcement and detection works.

CHAPTER 6

Development Enforcement Technique for Security Violations

Objectives

- *To define an enforcement system and its associated technique*
- *To expose our enforcement technique based on web service*
- *To propose decision and action architecture for violation activity*

6.1 Introduction

The integration of openness and distributed system technologies can improve security over the systems. Due to the widespread diversity and complexity of computer infrastructures and their associated systems, it is difficult to provide an ultimate secure system; therefore, numerous security systems have been developed. But even then, computer systems are vulnerable to be abused by insiders and penetration by outsiders, as shown by the growing number of incidents reported in the press. To close all the security loopholes of today's systems is simply not feasible and no combination of technologies and systems can prevent legitimate users from abusing their authority in a system; hence, attempts to keep the environment safe is viewed as the first line of defence.

As previously mentioned, policies (Hedi Hamdi, Adel Bouhoula et al. 2007) are widely diffused in networking services and information systems viz. security, management and QoS and they present promising and adoptable solutions for securing a wide range of Web application systems. Still, the adoption of a policy-based approach for security requires an appropriate specification and enforcement tool. The major remaining problem for a security mechanism is how to state, specify and correctly enforce its security policies.

An essential issue for the Internet and a networked information system’s protection is the specification and enforcement of a security policy. A security policy-based approach will describe its tools indicating how actions, e.g. permissions and obligations, can be passed. In addition, it will also describe how the enforcement model that supports the dynamic configuration of the system meets the security requirements of its policy.

This research, therefore, mainly focuses on how to identify systematically the correct policies instead of manually configuring them and how to enforce automatically security policies in distributed systems. Hence, a proposed approach is presented to overcome these issues.

6.2 Enforcement system

The main aim of the enforcement of a security policy is to protect and maintain a system to keep it fair and in a safe environment. The enforcement system has a set of wide ranging objectives, rules and activities but its basic role is in the area of regulatory control in order to meet statutorily based rules for the enforcement of security policies.

The purpose of an enforcement action is to achieve compliance from a user or consumer to provide protection and satisfy the enforcement policy's system.

The enforcement system is a means to provide mechanisms to the system which can ensure that its policy specification is not violated by the system's own execution. The standard model for policy enforcement that is used by most industrial enterprises approaches is defined in the ISO standard.

Most policy models and languages focus their discussion on their enforceability on the PDP. They show that the answer to an access control request is decidable and tractable. This is typically achieved by restricting the language/model to constrained data log programmes, which allow evaluating requests in polynomial time with respect to data-complexity (H. Janicke 2007).

6.3 Why should the policy be enforced?

The objectives of the enforcement of a security policy and its implementation procedures are to improve and control the service provided in its environment. Any services that do not adhere to the system policy shall be warned and compelled to establish the existence of facts in order to ascertain compliance with its regulations:

- In the interests of our model's security against intrusions.
- To prevent and detect threats and intrusions while the model is operating.
- To investigate or detect unauthorised use of our networked systems.
- To secure effective system operation.
- To define standards and criteria for different enforcement actions.

6.4 The Enforcement Technique

Several years ago, designers of computer security systems have developed automated system tools to analyse computer system audited data for suspicious user and system behaviour.

Technical scenarios in areas, such as, automotive or production systems will increasingly come to consist of a wide number of components co-operating in potentially unlimited and dynamically changing networks to satisfy the functional requirements of their execution environment. Due to the high complexity of these systems, it will be impossible to design explicitly the behaviour of their components for every potential situation that may arise. Therefore, it will be necessary to leave an adequate degree of freedom allowing for self-organised behaviour. Developing and designing the enforcement technique has become an urgent global priority, promising economic, financial, education and military benefits, the enforcement technique should maintain the integrity of endpoint devices and the enterprise network as a whole. Administrators can enforce all critical areas of endpoint security, including network access privileges of all consumers/users, devices and applications. This precise level of control prevents unsecured devices from serving as an entry point for a worm or hacker attack.

Enforcement technique (H. Janicke 2007) provides the bridge between policy specification and primitive operations e.g. device configurations, that are available in any given networking systems environment. The major challenge to enforcement techniques is that the accurate mapping of policies to primitives depends critically on the differences in the semantics between policies and primitives.

6.5 Detection Architecture

Certainly, having a policy means owing to enforce that policy. Wherever policies are implicit, an enforcer usually enforces them by working according to its policy or the regulatory requirements of the security’s infrastructure. A major purpose of a policy is to express the requirements at a high level of abstraction while hiding the details of the implementation that is necessary for their enforcement. Moreover, policies, which usually manage the behaviour of services e.g. security and QoS are becoming more popular methods for the dynamic regulation of web service environments. Policies can be logically divided into two main types (F. García, G. Martínez et al. 2005):

- Permission policies: concern those actions and accesses that entities are permitted to perform.
- Obligation policies: concern those actions and states that entities are required to perform.

A main goal of the policy-based approach is to allow network, service and application controls which are managed at a high abstraction layer. Using a policy language, the administrator disseminates rules that explain domain-wide policies, which are independent of the implementation of the particular network node, service or application. The policy based-service architecture provides support to transform and distribute these policies to each node and thus enforce a consistent configuration in all the elements involved by the enforcer. This is a prerequisite for achieving a means to dynamically constrain and control the behaviour of a system without human intervention. In the web application systems security field, a policy (i.e., security policy) can be defined as a set of rules and practices that describe how an organisation controls, manages, protects and distributes sensitive information from several different levels (Verma 2000).

6.5.1 Enforcement of Decision and Action

As aforementioned in chapter four concerning the enforcer architecture, we will here elaborate upon this architecture. Figure 6.1 below illustrates a typical action architecture that collates and controls when actions are applied in respect of any deviation that occurs in the services of the system. This architecture defines the ability to allow better visibility and agility and tackles the sticky cases of such contaminating entities. This section will describe how the enforcement of decision and action will take place, and how these steps help to assess the capability of the policy enforcer.

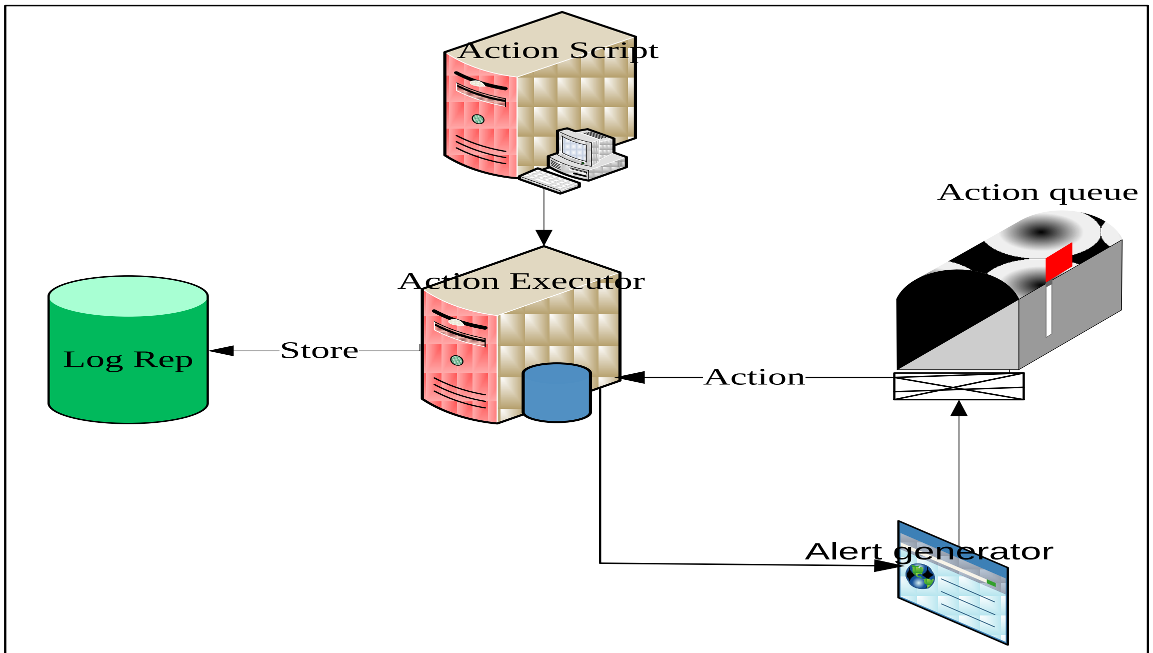


Figure: 6.1 Action of enforcement phase

6.5.2 The Decision stage

As previously mentioned, the policy enforcer (PE) is an online tool that detects violation activities in the system and their associated components. In fact, the enforcer model generates separate processes and runs each process parallel to observe closely any ongoing activities of services to the system domain.

The term ‘decision’ means making a judgment about whether there is a deviation at resource level. Decisions can be taken by comparing a resource’s legal state with its current state with regard to its terms, such as, privileges, behaviour and availability. At this stage, the enforcer obtains a policy from memory and observes on behalf of a resource, in association with the policy when there is violation activity. The enforcer then determines what actions are applicable and to what part.

As described in detail in chapter four, we have illustrated an algorithm at the decision stage that includes a collection of episodes which perform and examine a policy where (P) is the policy.

6.5.3 The Policy enforcement stage

A precise characterisation should be given for the class of security policies that are enforceable by mechanisms that work by observing system action executions and automata. This specifies exactly the class of security policies involved in enforcement. After detecting a deviation at its decision phase, the policy enforcer begins its enforcement, which is based upon the deviation itself. Policy enforcement is a set of executions of the action part of a policy in the case of a deviation. Each policy is defined by keeping a sensitive resource in service domains in mind. If there is a resource deviation in terms of the resource, such as, privileges, behaviour and availability, the action part of the policy is executed. The action part of each policy is a collection of scripts, which have a pointer to indicate an association with them. Therefore, the enforcer begins its enforcement by tackling and then triggering on the violation. Throughout the enforcement, the enforcer sends an alert message to the system’s administrator, which consists of event details, which are sent to the log repository for further diagnosis. Moreover, the enforcer runs a collection of scripts to repair any damage.

6.5.4 The Action stage

In figure 6.1 above, the action stage will execute and remove any deviation that was addressed by previous stages. The action manager receives a policy and stores it in a queue after detecting a deviation via the decision phase of the enforcer. A separate independent process or thread continuously executes queued action parts of policies. The action part is a set of a collection of scripts, which have a pointer to indicate its association with it. The action executer of the action manager simultaneously obtains a policy from the action queue and the collection scripts which are associated with it and fetched from the memory. The action executor then proceeds with its execution.

The collection of scripts has three method functions. The first function is to produce an alert message and then send it to the administrator. The second one is to reinstate the damage of the legal values of such resources. Finally, the third one is to send the event details to log-rep for more diagnosis.

6.5.4.1 Action stage diagram

Figure 6.2 below shows a typical sequence dialogue of manager action operations. It presents a diagram for receiving and responding to infected services. Firstly, the action manager receives an infected service and then the alert message based on applicable policies is added to the action queue. The message is then sent to the action executor after applying the collection of scripts. Finally, all event details are stored in the log file for more diagnosis.

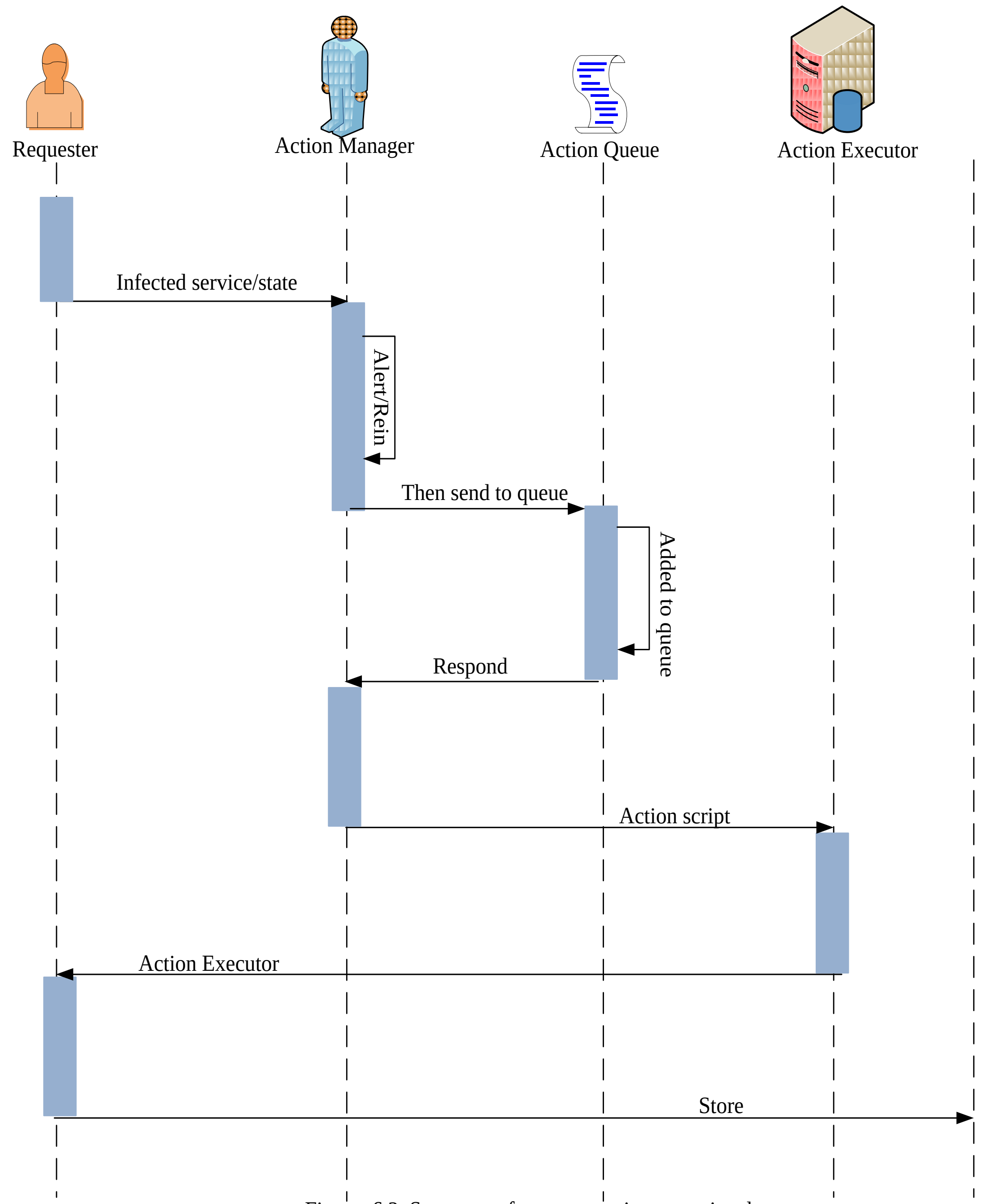


Figure: 6.2 Sequence of manager action operational

6.6 Summary

This chapter has addressed the issues surrounding the capability for controlling and reducing violation activity. The openness and heterogeneity of most system environments makes security a complex issue. Taming and managing this kind of environment are now highlighting security rather than its traditional issues. Our objective was to define end-to-end security. Consequently, this chapter proposed and developed an enforcement approach that protects and secures our system.

This enforcement has two parts: the policy manager (PM) and the policy enforcer (PE). They are introduced in detail as special objects that mediate the access to interfaces under their protection. We have defined for each enforcement mechanism and its associated methods, the abstraction over which the enforcement of the policy is defined. Furthermore, action architecture was developed, which enables the enforcement technique to detect, enforce and then take appropriate action on deviations.

The approach that we adopted is relatively straightforward and flexible. It is suitable for many organisational environments that use similar enforcement systems, in particular those where their systems are a facet of their security infrastructure. Organisations, such as, education, military and business that regularly retain highly sensitive information is considered prime candidates.

The proposed approach mitigates the threats by specifying the condition of the corporate security policy under which information can be exchanged during the model's activity. This will show and provide great potential for future extensions ranging from the combination of access control policies to property checks at the abstract level.

Policy enforcement acts on the potential risk and ensures adherence to our model's rules and granted security policy settings. The various policy enforcement mechanisms that have been implemented supplement rather than replace existing operating system security mechanisms.

The proposed policy enforcement is fragmented into two main parts:

- ✓ policy manager (PM)
- ✓ And policy enforcer (PE).

The PM logs event information provided from the policy enforcer and reinstates the default policy whenever it needs to. The PE is a collection of kernel-resident policy enforcement mechanisms that mediate actions to resources and impose the default policy.

In the next chapter, we will develop a prototype to demonstrate and validate the feasibility of the observation approach. It should span the possible application space of the targeted application domain.

CHAPTER 7

Prototype Implementation

Objectives

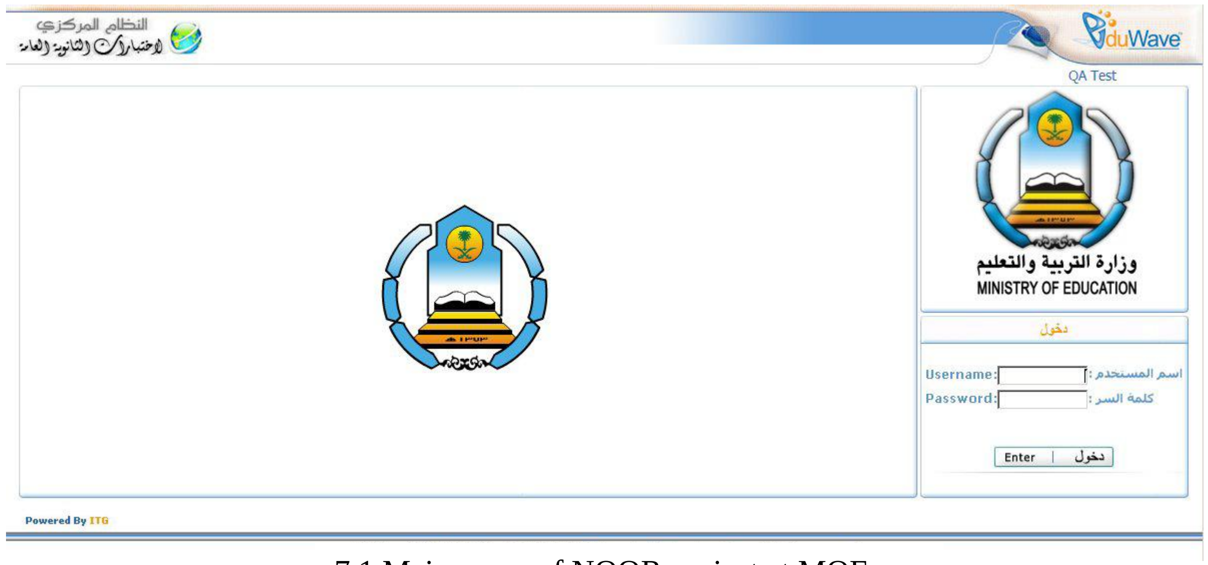
- *To present the architecture of the eStudent system*
- *To develop and implement the eStudent system*
- *To describe the implementation of the eStudent system*
- *To integrate an algorithm method of Observation technique with eStudent system*

7.1 Introduction

The development of education in the Kingdom of Saudi Arabia (KSA) is evolving rapidly thereby increasing its capability and facility to deliver a high quality of service to the public. The government of the Kingdom of Saudi Arabia has developed education as a first priority for the nation. It has charged the Ministry of Education (MOE) with the task of revolutionising, modernising and expanding the Kingdom' education system and increasing the number of students of all ages. The MOE, therefore, has overseen the introduction and fast growth of IT in education. This has introduced many valuable opportunities for teachers, students and their parents in the education sector. As a result, the MOE now supervises a huge number of students (both boys and girls) of all abilities together with their teachers. The number of students in the KSA is approximately five million. They are registered at different schools and educational institutions in a number of subject areas. Some students are of Arabian nationality, such as, Egyptian, Jordanian, Palestinian, Kuwaiti, UAE, Qatari, Omani, Syrian, Moroccan, Iraqi, Libyan and Sudanese. The education system in Saudi Arabia is similar to the American education system. The Kingdom's system involves a student with six years of primary, three years of secondary and three years of high school education. The MOE organises and manages education strategy and its associated system and provides the service to the public. Several years ago the MOE started to introduce the use of information technology into its departments and schools, which has made the education system easier for teachers, students, parents, staff and researchers to access and use.

7.2 The NOOR Project

MOE is rapidly adopting IT which has influenced the direction of software design. This is because of the constantly changing and ever-increasing demand for education services based on the Web. The NOOR project is one of most used projects in the KSA and in the field of education. It has a high budget to develop and perform it for serving a huge number of students in different schools in the KSA. This project has various software applications for example, school management, student clinics and student marks. In this chapter, we will make a simulation of the NOOR project and adapt a small part to design a prototype to accomplish the proposed approach. Figure 7.1 below shows the main menu of the NOOR project. Thus, we have adopted one part of the applications of the NOOR project and have applied it to exhibit and validate the feasibility of our proposed approach.



7.1 Main menu of NOOR project at MOE

However, the development of IT in the education sector will introduce many benefits. Some advantages of using advanced technology are to join the MOE and its staff, students, parents, teachers and users with more advanced and powerful applications; electronic education records can be retrieved more easily; industries are enabled to be able to develop information sharing technologies within the bandwidth education networks. All users expect that a new paradigm in digital education solutions will offer new challenges to the underlying user teaching, care, communication and network infrastructure.

Information technology (IT) can make a significant contribution to education by improving the quality of its service and efficiency. Research indicates that the education industry is always willing to invest in IT in order to become an efficient accurate and high quality service. The complexity of implementation and its unbearable cost are the main barriers that confront IT applications in the education sector.

This chapter will describe a technological approach that delivers learning services online by using a distributed system as well as demonstrating how teachers, students and their parents can use and interact with high levels of technology over the network.

Our objectives in this chapter are to provide an assistant tool-support for the analysis of policies. This assistant tool aims to allow for the early and frequent validation of policies based on an observation approach, which is integrated into this development software. In addition, the aim is to test whether a policy captures the original intent when enforcing our policies in the system.

7.3 Prototype Framework

The eStudent system is a whole development system designed and implemented to facilitate and solve many of the issues at the MOE's interactions, as well as supporting students and teachers to use this system to facilitate interactions between themselves, parents, school management as well as the MOE. This system will systematically work to reduce time and cost when a student requests a service.

With the rapidity of high-speed network communication, vast information technology and huge digital storages has reduced time consuming to access more convenient learning facilities. With concern to IT in education this part will show many advantages for its improvement and associated parts for example, students and teachers. This system manages and provides access to a wide variety of services e.g. schools. Access into the system is via different component technologies, such as, the web, mobiles and PCs. Our system is processed by components that have been built into various standardised education applications along with a standard communication protocol to communicate between the system's components. Security and the privacy are enforced with the help of a modern security mechanism, which has been described in previous chapters.

The proof of our research's concept system is elaborated in this part of the chapter and it introduces a novel approach to education management in IT.

7.3.1 System Design

All expected requirements of the eStudent system should be applied and implemented to be satisfied.

The major design principle upon which the system is based is extendibility. The eStudent system is required to provide and support all marketplace requirements while containing the facility to adapt to new ones as the education sector is a rapidly evolving industry that introduces new initiatives daily.

Firstly the main inputs and outputs are processed as manipulating inputs. They generate the required output, which is the primary concern of any system. For eStudent, there is variety of inputs both defined and undefined. As mentioned, the system should be able to process and manipulate unidentified inputs since extendibility is enforced. Table 7.1 catalogues defined inputs and their types.

Input	Students basic information	Students records
Type	Numeric/Text/String	Documents/files

Table 7.1 defined inputs/types

A database can be considered as the core of our system. The major aim of a database is to store all the data that relates to eStudent. As assumed in the table above, the database has to store all the expect data formats as described above. It is designed for storing and retrieving data efficiently in order to manage a huge number of users, e.g. students and teachers, at once. The main requirement of the database is concurrent processing of data, as student information can be manipulated from anywhere simultaneously. Because the eStudent system can be accessed from remote locations via Web services, PL-SQL stored procedures are used to provide for the huge traffic usage that is expected. This is achieved by reducing network traffic and serving users with a quick response to data. MS SQL server is used for the database, which is one of the best database technologies that support all the aforementioned techniques.

7.3.2 eStudent Client Applications

eStudent is designed and implemented to provide different types for users from different locations facilitated by the most modern communication technologies, such as, ADSL, WAP and Wi-Fi. An important characteristic of eStudent is its ability to provide these technologies for users e.g. teachers, students, parents and staff members.

7.3.3 Observation system

As shown in figure 7.2 below, the Observation System is embedded to reinforce and control the eStudent system when any attack occurs. As described in detail in chapter four, the observation system has three main components: an enhanced observer, an evaluator and an enforcer.

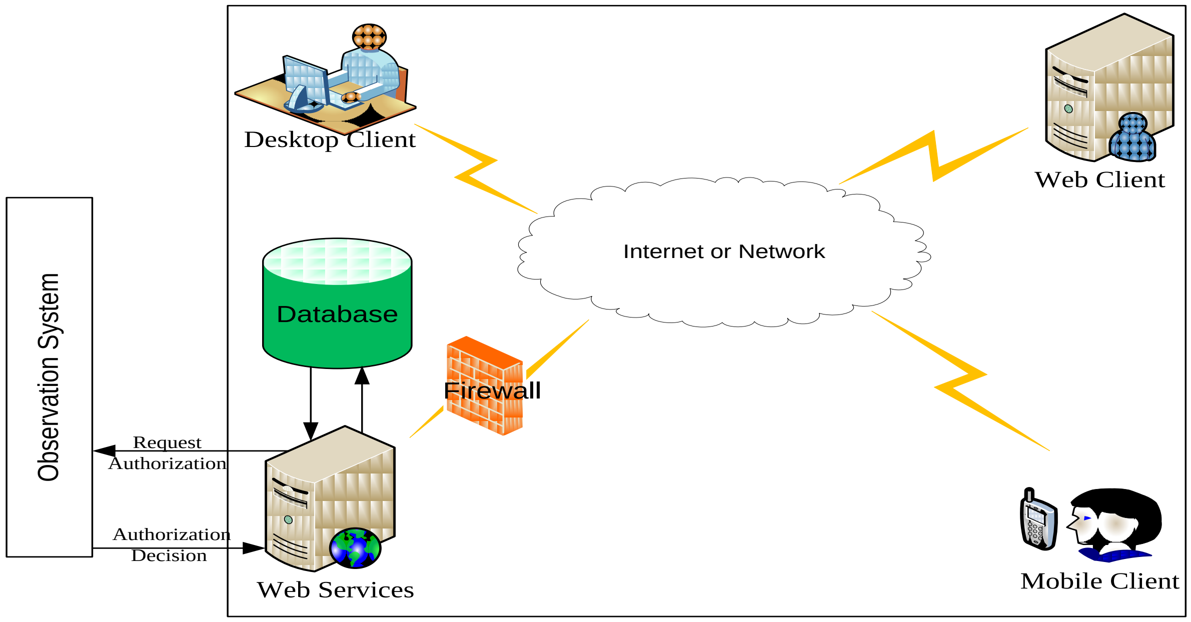


Figure 7.2 Framework of eStudent system

The observation system works as an online active tool that controls and increases security. It is traditionally used to increase the surveillance of the system. It has been found and used by several enterprises. In addition, it is implanted within the eStudent system to enhance its security. It acts by observing any interaction between services in the system. It is an additional tool that controls and minimises any leak of security. This tool is not rogue software, it implants within the eStudent system to trace all services.

Figure 7.3 below shows the relationship between the eStudent system and the observation system. It shows that the eStudent system works as middleware between users and the observation system. The observation system has been implanted to manage all requests that come from users, which are then examined before accessing the resources. These requests, which are sent by users, are services. The user may notice a delay during the process because a service may pass via all the observation's components. So, the system will respond when any deviation occurs during sending or receiving.

The observation system is not part of the eStudent framework; it supervises the system by checking incoming services from users and when they interact with each other. As shown in algorithm 7.1, a scenario of the observation system begins as follows. The student requests an appointment with a tutor concerning a course or assignment. The system will respond to a student’s mobile phone or email that is registered in the profile. The tutor should receive an email about that request. eStudent has the ability to find the optimal services that meet the user's demands, therefore, it will search in the tutor’s school and database to find his or her timetable and appointment and then respond to the student.

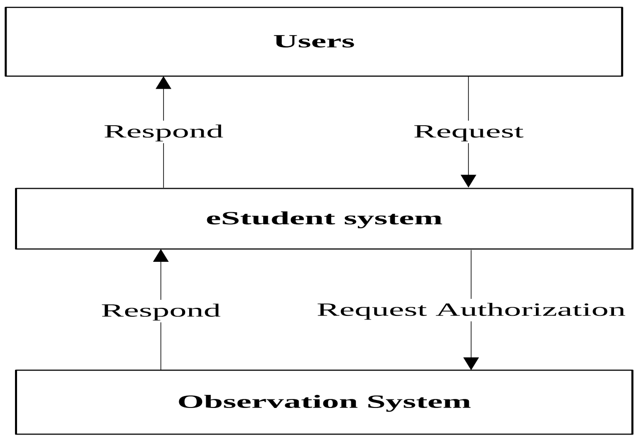


Figure 7.3 Framework relationship between eStudent and Observation systems

To realise the system requirements, as seen in figure 7.2, eStudent defines three main user terminals, which will cover almost all technologies. Those client interfaces are stated and described blow.

- *Desktop Client*

The first one is the Desktop Client, which is a software programme that is installed on a PC. Desktop client will provide users with an interactive user-friendly interface to use eStudent services. Desktop clients are recommended for use in schools, offices and other educational institutions.

- *Web Client*

The second one is the Web Client, which is simply a website that permits users to use eStudent services from anywhere and is recommended for use by those who do not use a permanent PC.

- *Mobile Client*

The third one is the Mobile Client, which is a small software application that runs on small devices, e.g. mobile phones and PDAs. Mobile Client is recommended and useful for portable and roaming users. Users, therefore, can use eStudent services via their mobile phones when they are travelling or roaming but based on a Wi-Fi connection.

The most important component of the eStudent system is the Web service. All the eStudent components and services are implemented by Web services. All end user interfaces are involved with Web services in order to make requests. As stated in the figure 7.2 above, the major component linked to web service is the database. The main features of the research are inter-operability and extensibility. These features are enforced by the Web service. The communication standard of the web service here is via SOAP, which is the optimal service in different client technologies to access eStudent services.

The main concern for our system is security, which is integrated during the development of eStudent, because educational records are considered highly sensitive information. Security, which was described in previous chapters, will be enforced in this system by the enforcer model based on its policy.

Another concern is the privacy of the users. A huge amount of data is provided by eStudent and its services; therefore, it is felt that users would be reluctant to disclose important private information, e.g. names, addresses and DOB. Furthermore, interlopers may try to access the system to obtain and modify the user's records or information. To solve and overcome unauthorised access, this system has a security implementation based on our observation system, which helps to preserve the system's integrity. This defence mechanism explained in detail in the last chapter how violations can be detected and what response to make. A user must register before using eStudent and then they are able to use the system. In order to register, users must provide a name, username and password each time they login to the eStudent system. Web services have to use a high-level of encryption to communicate with a client's software. As shown in figure 7.4 below, the use case diagram is drawn to state all users' interactions with the system.

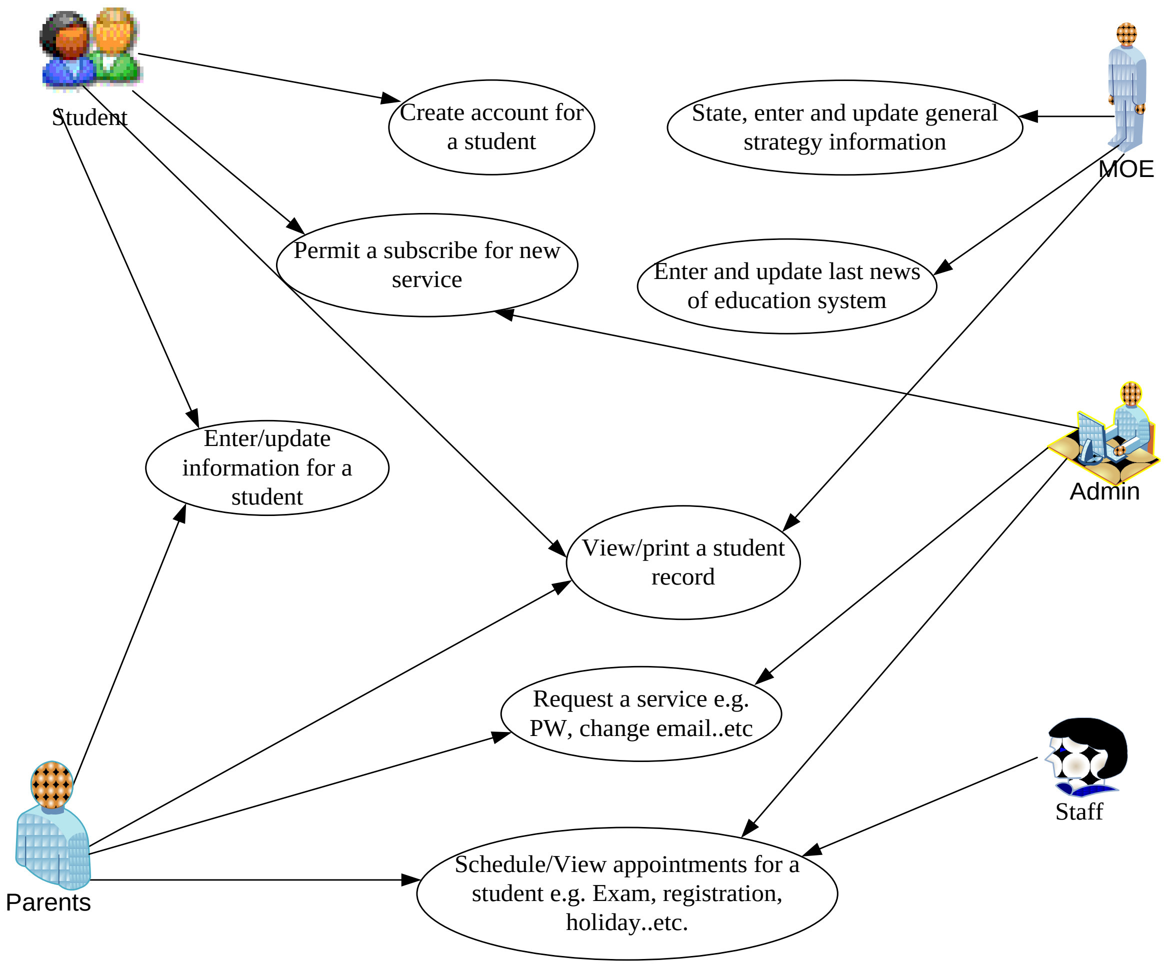

Student
Create account for a student
State, enter and update general strategy information
MOE
Permit a subscribe for new service
Enter and update last news of education system
Admin
Enter/update information for a student
View/print a student record
Request a service e.g. PW, change email..etc
Staff
Parents
Schedule/View appointments for a student e.g. Exam, registration, holiday..etc.

Figure 7.4 Use case for students and other relation

7.3.4 Application Components

The application components of the eStudent system allow access to its data via multiple application component types; therefore, users are able to access the system via many devices regardless of their location.

The system's main application is developed and implemented as a desktop application. Users e.g. teachers, students and staff can use the desktop application within their offices, e.g. in schools and in the MOE, and if they install the application on their PCs or mobile computers (Netbooks) based on accessing the Internet.

Moreover, the two applications, the desktop and web-based, are designed and implemented with the same core forms. The ability of the web based application is that it can be accessed and used via any machine using an Internet browser. The web based application will be designed by ASP.NET and is hosted on the Internet Information Site (IIS) running behind the firewall as stated above in the figure 7.2.

Furthermore, the mobile client application is designed for using on smart phones using and running on Microsoft Windows platforms. The application will be designed using the .NET framework environment. This type of application will serve and enable users, e.g. students, to access and request services via their mobiles. The limitation of this application is that available information is limited when comparing it to the desktop application. Therefore, all these different categories of applications will be accessing the same web services to obtain information.

7.3.5 Database Design

Figure 7.2 shows that the eStudent system is linked to the database which is developed to store and retrieve data efficiently and accurately.

As mentioned earlier, the education part in the KSA is a great industry that is founded and provided by a number of institutes in both government and private parties. The capabilities of the eStudent should cater for all these parties with the ability to address their demands. The implementation of eStudent is carried out in different stages, as it is difficult to implement a collection of application systems at one time. From a technical viewpoint, the infrastructure needs to be in place first before eStudent can be efficiently installed.

Essentially, the servers need be set up in a suitable location e.g. at the MOE, and these servers should have their physical security. The server requirement is database server and Web server. For the database server a MS SQL server is used while the Internet Information Services (IIS) server is used as the Web server.

7.3.6 The Algorithm of observation method

Pseudo code is a process of compact that illustrates a high-level description of a computer programming algorithm. It is informal and uses the structural conventions of a programming language with basic explanation, but is proposed for human reading rather than programming or machine reading. As illustrated below in algorithm 8.1, the pseudo code of main observation method() describes how services can be observed, evaluated and then enforced.

This algorithm has shown the compact of implementing the main method of the proposed technique which is impacted on the observation system. It is embedded to underpin the security. This algorithm is not part of the eStudent system but is immersed to act when it is requested.

This algorithm typically omits details that are not important for understanding the algorithm, such as variable declarations. The programming language is detailed with descriptions, where convenient, or with compact mathematical notation. The purpose of using pseudo code is that it is more easy to understand than conventional programming language code. It is also sketching out the structure of the programme before the actual coding takes place. In the algorithm 8.1, the observation method() consists of three methods and they are observer method(),evaluator method() and enforcer method(). Firstly, the observer method() will receive two things, a number of services and the interactions between services which are: *getService_no() and getAll_Servcies()* and these tasks should obtain the number of services that come in SOM and their interactions in services. Then observer method() will observe these services and sends them to the evaluator method() as shown in the phase (i) and (ii) respectively.

Secondly, the evaluator method() should receive the outcomes of interactions between the services that are observed. The evaluator method will examine and assesses the behaviour and history of the services that are observed by the observer, *getReturn_of_Services()*,whether these services have ‘deviation’ , and then sends the report by *Report[P]()* as shown in the phase (iii).

Thirdly, the enforcer method() will receive the report to detect, re-evaluate and then action as shown in the phase (iv).

Algorithm of observation method()

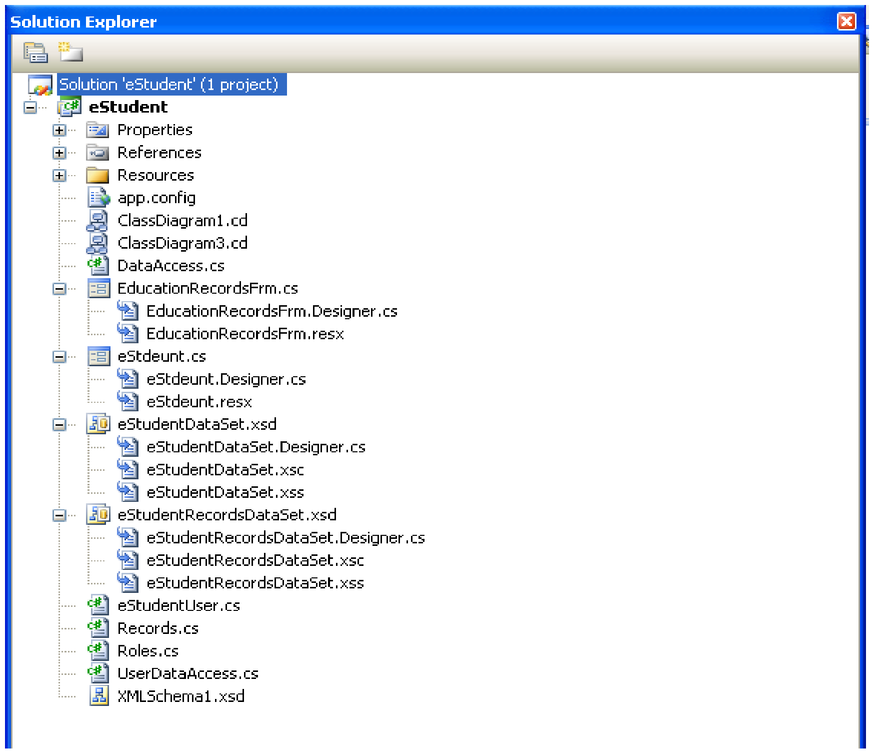
```
Variables=counter, Interact_of_services[A], List_of_services[L], Return_of_services[R], Report[P]
Begin

        For all services ∈ All_Services
          All_Services←getAll_Services

do
            do    Counter ←getServices_no                                           (i)
                  List_of_services[L]←All_Services
                  For i=0; i < counter ; i++
                      Read(list_of_services[L])
                  End for

        // observer method ()
            do    Interact_of_services[A]← getAll_Services                          (ii)
                    For i=0; i< counter ; i++
                        getRotate Interact_of_services[A]
                          Read(Interact_of_services[A])
                  If anObservable is registered with observer, add anObservable to the list of observers
                      Otherwise, register anObservable and add anObserver to its list of observers.
                        If found interaction between any services ∈ All_services
                        then   Insert Interact_of_services[A]  into Return_of_services[R]
                                  .....

                        else if   no interaction found with ∈ All_services == ∞
                                          ......
                    End for

        // evaluator method ()
            do    While Return_of_services[R] ≠ Null                                (iii)
                    For i=0; i < counter;  i++
                        Read(Return_of_services [R])
                          Evaluate interaction into Return_of_services
                          If  (Return_of_services [R]) ≠ satisfy

                          else if   (Return_of_services [R]) ='deviation'

                                then   Report[P]← getReturn_of_services [R]
                      End for
                  End while

        // enforcer method()                                                        (iv)
            do    For i=0; i < counter; i++
                      check(Report[P])
                        if (Report[P]='deviation')
                          Detect 'deviation' in (report[P])
                            then   Re-evaluate (report[P])
                                         Still has'deviation'
                                   else if   Report[P] is == 'deviation'
                                             then   Report[P] ≠  satisfy
                                                    get.remove('deviation')
                  end for
End
```

Algorithm 7.1: the main observation method ()

7.3.7 Development tools

The powerful and capable Microsoft .Net framework and its features act as assistants and are there to use and design the whole system. Microsoft .Net framework offers many features to develop our system efficiency. Visual Studio 08 and above is the Integrated Development Environment (IDE) used to design this whole system. In addition, MS SQL server is the database server that is used.

For the windows application, the system's solution is constructed as shown below:



```
Solution Explorer
Solution 'eStudent' (1 project)
  eStudent
    Properties
    References
    Resources
    app.config
    ClassDiagram1.cd
    ClassDiagram3.cd
    DataAccess.cs
    EducationRecordsFrm.cs
      EducationRecordsFrm.Designer.cs
      EducationRecordsFrm.resx
    eStdeunt.cs
      eStdeunt.Designer.cs
      eStdeunt.resx
    eStudentDataSet.xsd
      eStudentDataSet.Designer.cs
      eStudentDataSet.xsc
      eStudentDataSet.xss
    eStudentRecordsDataSet.xsd
      eStudentRecordsDataSet.Designer.cs
      eStudentRecordsDataSet.xsc
      eStudentRecordsDataSet.xss
    eStudentUser.cs
    Records.cs
    Roles.cs
    UserDataAccess.cs
    XMLSchema1.xsd
```

eStudent includes four individual development application components. These are:

1. A windows desktop application
2. A web application
3. A Web service
4. A mobile application (Under development)

All these application components are developed and implemented individually in separate solutions by MS Visual Studio. The web service and the web application should be set and hosted by the IIS server. Afterwards, the database should be attached and added to a SQL

Server by using the SQL Server Management Studio Express and then the connection parameters are set on client applications.

The core functionality of the system is controlled by the eStudent Web service. All client components can access the Web service to achieve all the services that are provided. For instance, if student wants to request a service e.g. complete a test on a subject, the student can simply login to the web application or mobile application and place the request via the web service. Then the student should receive a response about that particular request.

As shown in figure 7.5 below, users are required to enter their user ID, name and password when accessing both the applications on desktop and the Web.

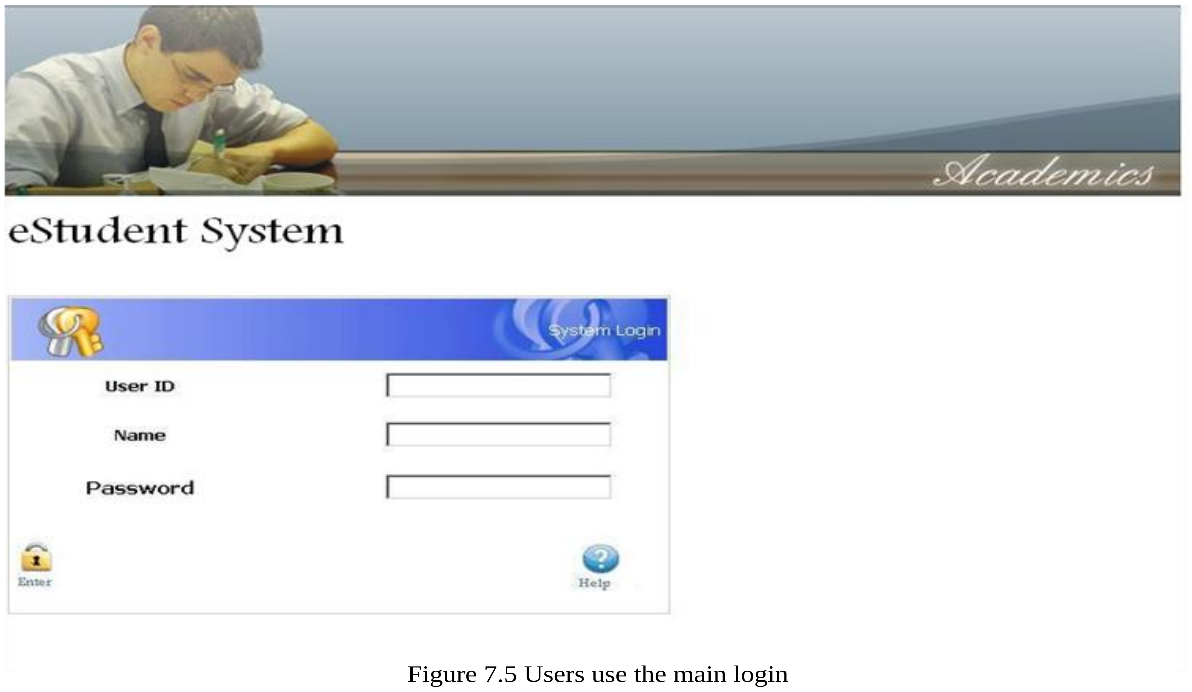


Figure 7.5 Users use the main login

7.3.8 Mobile Application

The mobile and web applications show the booking functionality which allows the student to book and request a service from their school e.g. book an appointment with a tutor. This application is based on a Wi-Fi connection. After that, the student should receive confirmation of the request. This confirmation should show the student's ID, student name…etc, as follows and an email will be sent.

7.3.9 Web and Desktop Application

The eStudent desktop client application works as a Web and mobile application with additional features. This application should be installed at schools for use by teachers, students and members of staff. In addition, it presents all administrative functionalities, such as, managing student information. Students can add, view and update their information. In addition, this application allows students to submit and retrieve any enquiry, such as, a request for a subject mark or to book a test.

7.3.10 Testing

Benchmarks are formulated to mimic a particular type of workload on a component or system. The computer programmes are used for compiling some of the benchmark data. The Computer Language Benchmark's site includes a large number of micro-benchmarks of reader-contributed code snippets with an interface that generates different charts and tables that compares specific programming languages and types of tests.

At this step, and after the development of the system is complete, it is essential to test it thoroughly before it is deployed. This is because the system will be used in education where availability and reliability are the main features, and which users expect from the system. Consequently, overall testing of the system should be conducted. To ensure all the functional and non-functional requirements are checked and functioning properly, testing is conducted in several stages throughout the entire system's development process.

The initial test uses unit testing where the consistent functionality of each code segment is checked. Unit testing is also accomplished by the designer and completed before the end of the development process.

A number of unit testing tools are available to assist designers to ensure that unit testing is completed perfectly. Each of these tools is developed by appropriate software development technology.

In addition, the next stage of the testing process commences. The overall testing of the software is carried out component by component or part by part as eStudent is shaped by the integration of several application components e.g. Desktop client, Web Application Mobile Client as well as Web Services, hence the span of inputs and output is relatively large.

The major aim here is to ensure that the system comprehensively meets expected functional and non-functional requirements by defining potential faults. All the faults and suggestions should be reported back to the development process.

Since the system is mostly tested component by component, it cannot be considered a fully tested distribution system, because the system as a whole is based on the proper interaction between components and if these interactions are out of order then particular components could be considered useless.

To overcome this, the next stage of the system's testing process is integrated testing, which focuses on the correct interaction between components so they fulfil their requirements and duties.

The last stage of the testing process involves a special test, called a user acceptance test to ensure that the user is fully satisfied with the system that has been developed.

As a result, there are two main types of testing methodologies, which are applicable for any testing stage discussed above and they are used by several enterprises as white box and black box testing.

- ✓ **Black box**

In this approach, the tester acts as a typical user to test the system. Knowledge of the internal code level is not required for the tester. This is because only the system's functions are being tested. In a black box test, it is not required to know any details of the code.

- ✓ **White box**

For white box testing, the tester must know the code levels for all the internal implementations.

Testing is carried out by concentrating on all code paths and their desired functions individually in more detail. In addition, all the components are tested altogether (also called integration testing) to confirm that components are working together as required.

7.3.11 Discussion

In general, the research development system is introduced for use by public bodies, such as schools and education institutes. They are developed and deployed for use by educated people. The major purpose is to ensure that all requirements of the system satisfy the user's expectations. All users of the system must possess an account with a user ID and password to login into the system. Web client is expected to be used by students and parents. Mobile client is intended for use by students to make requests or update their information. The functionalities of the mobile application have some limitations as it runs on a mobile phone or PDA, which has limited resources and is based on a Wi-Fi connection.

When the system is more stable and comprehensive, the second phase of implementation will start. The knowledge base of eStudent is designed to reach a satisfactory level after the first phase of implementation. Many organisations, in particular the education sector, become involved at this stage. Students, parents and teachers are able to interact with each other in real time using the facilities that are provided by the eStudent system. It is essential that the web server is readily available and reliable. The server should have the capability of being able to cater for a wide range of user requests and the capacity to accommodate a large flow of data. Replicating and restoring servers are suggested to solve sudden hardware and catastrophic failures.

Further, eStudent is proof of a concept that was developed and implemented to offer a facility for teachers, students, institutes and other relevant parties to perform education activities collaboratively. This proof is validated by the proposed approach that used and applied the observation technique.

Lack of access to education information is the major issue in the field, and is addressed by eStudent's technological approach. eStudent provides global access to education information via the World Wide Web (WWW) and the Internet, which enables users of the system to receive accurate, reliable information in real time.

Our system is developed and implemented by using the best software development technologies and methodologies, which enforces its dynamic expansion while it is being used. It is simply ready to add new features as an organisation evolves. Accessibility to the system is increased by implementing several client components and users can use those components as required. The system deals with highly sensitive data, therefore, the integrity and security of the system is very important and security features are embedded and implemented via the observation system along with the system's development.

In addition, The MOE and associated departments as well as a wide range of schools have used it and have provided good evaluations and valuable feedbacks. The compilation of education people and the IT sector will contribute to the community of students, teachers and researchers in the future.

7.4 Summary

In this chapter, we considered the development and implementation of the eStudent system prototype, which leads from the abstract specification of education to a concrete implementable code to ease using the facility among the MOE, schools, teachers and students. The eStudent system is a simulation of NOOR project.

We outlined and depicted the architecture of the eStudent system and embedded within it the Observation system to reinforce the security violations. The eStudent's implementation demonstrates the feasibility of the education techniques involved.

With rapid development in IT and the education sectors, collaborative and integrated education systems are required to support such technological management and administrative activities of information dissemination in the form of an education service. Surveys indicate that the majority of teachers and students are willing to use this kind of technology. In fact, developing and implementing a comprehensive education system contributes a range of capabilities to the education system in the country and for their teachers, students and parents.

Since many other aspects are influenced by implementing this system, students and teachers will consider eStudent as a valuable resource that enriches their lessons. Business sectors can also use eStudent to market their valuable products. For the smooth expansion of eStudent through various geographical regions, bilingualism is essential, which should be implemented as a future improvement.

In the next chapter, we will provide an evaluation of this prototype to demonstrate and validate the feasibility of our planned approach.

CHAPTER 8

Evaluation

Objectives

- *To present the analysis and evaluation technique*
- *To evaluate the prototype of eStudent system including Observation approach*
- *To define and implement the Observation methods*

8.1 Introduction

With the proliferation and revolution of the Web it is not only used as a research tool but also by many major sectors, such as, education, business and health. The Web uses business communication tool groups as new ways to communicate. Technologies, such as Web services, permit communication structures to be built easily and efficiently, giving even more interaction and participation. They support this process by using the information that is found on the Internet in new ways, even if only allowing users to subscribe to pouches of information. Services are directly notified when any new information is published and this should help to create complex business applications to control complex real-time offer chains. System architectures normally become more and more complex due to increasing functionality and by implementing additional features.

Hence, the analysis and evaluation processes are important issues; they divide a complex topic into smaller parts in order to obtain a better view and understanding of a problem. This will imply a common system view with comparable structures and similar modelling approach technologies. The overriding process of analysis in the eStudent system is to gain a greater understanding of the needs of users in order to fulfil their requirements. The main purpose of this chapter is to analyse and evaluate the observation system by examining the behaviour of services to obtain satisfactory results. It will focus on the value of the research study to the research community, researchers and students.

8.2 Evaluation

Evaluation is the process of assessing and examining a subject. It is based on criteria that rate its significant features. This evaluation will describe and explore how the eStudent system is valuable when it is integrated with the observation system. In addition, evaluation is a popular technique used in a number of different areas (Schmitter 1996). By applying expect criteria to discover an entity's strengths and weaknesses we can then decide how much or how little we value something by using judgment depending on measurements that have been predefined. The major advantage of evaluation is that it will state, whether the entity, in this case the observation system, performs satisfactorily.

8.2.1 Analysis the Prototype system

As previously stated in the previous chapter, the prototype of the eStudent system is a whole development system that was designed and implemented to solve numerous educational issues which is a simulation of the NOOR project in the MOE. It is intended that the system supports students and teachers in using its facility.

Therefore, in this section, we will present critical evaluation for the observation approach which is integrated with the eStudent system. This evaluation should provide explanations for the main aspect technique of the observation. It is used to validate and demonstrate the feasibility of the proposed approach, which showed that the methodology could work and produce impressive results to span the potential application space of the targeted domain.

8.2.2 Evaluation of the Observation system

As mentioned in chapter four, the observation system is responsible for actually changing the processes internal behaviour and structure by issuing new parameters or process configurations. A main goal of observation is to observe service behaviour to determine whether it complies with its intended behaviour when interaction occurs. The observation system observes the services and checks correctness by comparing an observed state of the services with an expected state of the services.

Hence, we have to assume some criteria that should be based upon when using the observation technique so, to implement our planned approach we have set and used a confederation system that has pre agreement from those observables which will allow the observation of the behaviour of incoming services by collecting data and then processing it. Thus, our assumption is, tracing and observing the interaction communication between services under confederation and then takes the decision based on their behaviour and history.

The big issue here is how do we ensure that some given security requirements are satisfied and enforced?

Therefore, the confederation scenario is based on the following:

- System's components are Web-services.
- These components are black boxes, designed/built by various vendors.

The "observation" allows to observe (partial) behaviours of those services and to construct complete behaviours that can be analysed and compared with security requirements. So we have two sets of behaviours: (a) the observed (partial) behaviours; and (b) the actual required behaviours.

Given these two sets can then be compared, it can be determined if the system satisfies the security requirements. If the two sets are the same then the system satisfies the requirements; otherwise it does not and needs to be detected.

Consequently, the e-observer observes these services when interacting with each other and then sends them to the evaluator system which assesses the behaviour of services to discover any deviation that is based on their history. The evaluator then writes to the enforcer system to make the detection, re-evaluation and then action. In a technical system, the observer observes the external environment via the sensory input as well as the internal behaviour of the low-level execution unit and is highly manipulated in several ways.

8.2.2.1 Behaviour

As mentioned in chapter two, behaviour is the way of responding to a system to the situation that has been found. It is a sequence of state of a system.

The behaviour service is a description of a sequence of state that specifies dynamic aspects of an entire system. It specifies the states and modes of the system that could impact with its environment. So, we will use and analyse the behaviour of the service to determine whether it obeys the expected behaviour.

8.2.2.2 The Proposed Technique

As elaborated in previous chapters, the e-observer technique addresses most problems and provides ultimate solutions that have been proven in similar contexts. The proof of the solution lies in its ability to provide a facility for the system in order to collaboratively perform and control any violation activities. The e-observer addresses these activities by a technological approach. The e-observer is responsible for appropriate observation and feedback. Thus, the e-observer observes behaviour through sensors by comparing the results with the expectations and decides what action is necessary to provide the best-known action through the enforcer system, but only after the evaluator has assessed the results.

The advantage of this technique is to deal with the e-observer technique over the Web environment, which makes it possible to achieve adaptability in order to improve system performance in dynamic environments by efficiently using the available resources and controlling any activities.
The observation behaviour itself is variable; hence, the e-observer influences the observation procedure, e.g. by selecting certain detectors or certain attributes of interest. Based on the aggregate results by the e-observer, the evaluator can benchmark the data with an objective function and knows what actions are best to guide the SOM in the desired direction by informing the enforcer model, which acts as a switcher to detect the violation. Our assumption here is, tracing and observing the communication between services and then taking the decision based on their behaviour and history. Hence, the big issue here is how do we ensure that some given security requirements are satisfied and enforced? The scenario of observation and its observables can be formed as the following:

8.2.2.2.1 How does the observer technique work?

- ✓ **Observer Registration**

Figures 8.1 and 8.2 depict the registration sequence; the e-observer invokes the register method on the subject, passing itself as a conversation or argument. Once the subject receives this reference, it must store it in order to notify the e-observer when a state change occurs sometime in the future. Rather than storing the observer reference in an instance variable directly, most observer implementations delegate this responsibility to a separate object, typically a container. Use of a container to store observer instances provides important benefits. With that in mind, the next action in the sequence is the storage of the observer reference denoted by the invocation of the add method on the container.

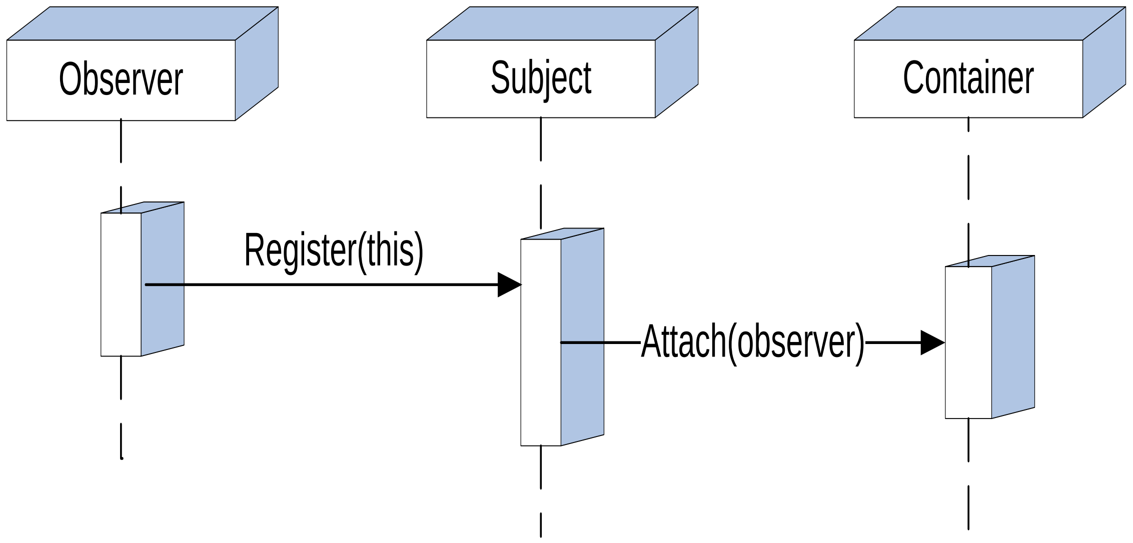


Figure 8.1 Observer can register and store in container

```
{
void Register(Object anObservable, Object anObserver)
    If anObservable is registered with the manager, add anObserver to the list of observers for
    anObservable,
otherwise, register anObservable and add anObserver to its list of observers.
```

Figure 8.2 Observer can add or remove objects

✓ **Observer Notification**

Figures 8.3 and 8.4 highlight the notification sequence. When a state change occurs *(Changed(state))*, the subject retrieves all the observers within the container by invoking the *GetObservers* method. The subject then enumerates through the retrieved observers, calling the *Notify*() method, which notifies the observer of the state change.

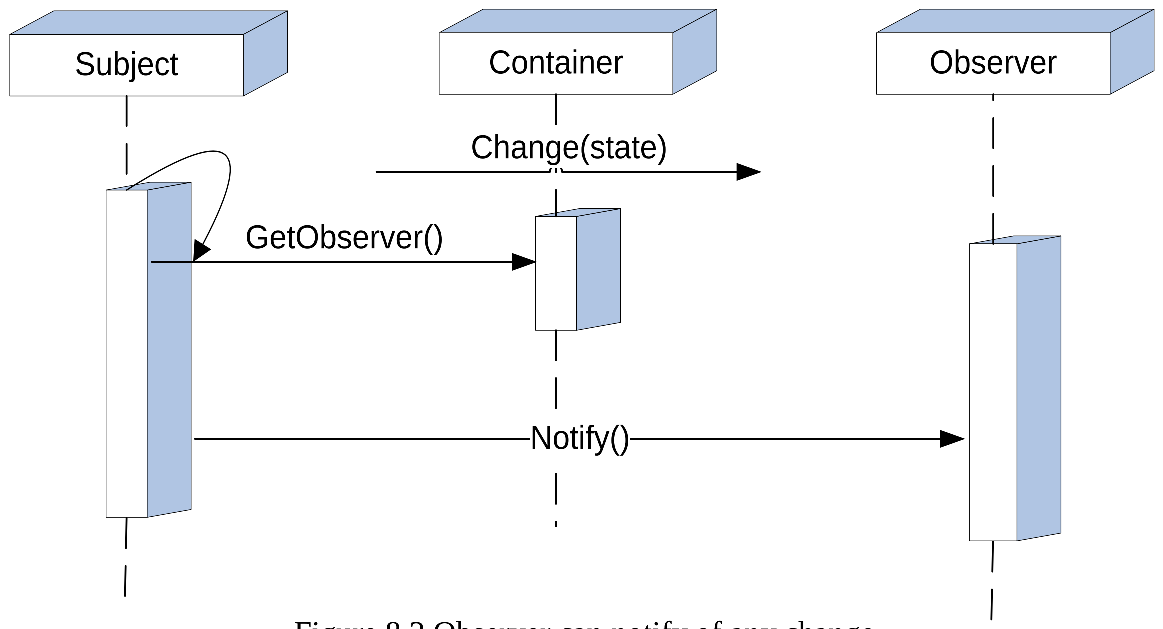


Figure 8.3 Observer can notify of any change

```
  void notify (Object anObservable, Object arg)
  {
    if ( anObservable s exists )
    { for ( all observers of anObservable ) // process in reverse order of registration
    { currentObserver.notify (anObservable, arg) // combine push and pull models
    place the current observer on the visited list
  if ( currentObserver is also an Observable )
call notify (currentObserver, arg)
          }
     }
 }
```

Figure 8.4 Methods notify all registered observers

✓ **Observer Unregistration**

Figures 8.5 and 8.6 present the unregistration sequence. This sequence is performed when the observer no longer needs to observe the subject. The observer calls the *Unregister* method, passing itself as an argument. The subject then invokes the *Remove* method on the container, ending the period of observation.

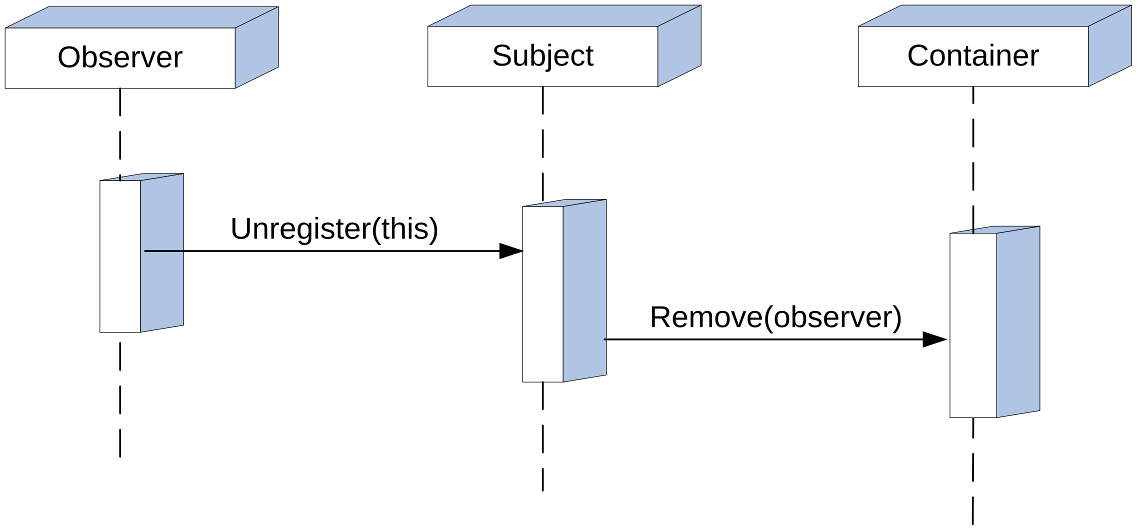


Figure 8.5 Methods deregister all registered observers

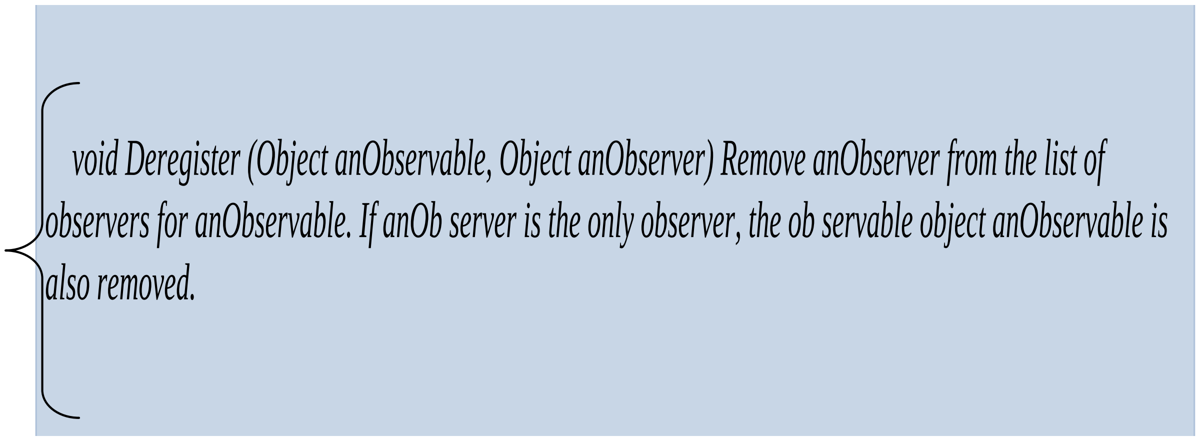

void Deregister (Object anObservable, Object anObserver) Remove anObserver from the list of observers for anObservable. If anOb server is the only observer, the ob servable object anObservable is also removed.

Figure 8.6 Methods deregister all registered observers

8.2.2.3 Validation of Observation system by FSM and AGG Tools

As shown in figures 8.7 and 8.8, the main role for using FSM is to facilitate the proposed approach of the observation system by complying with the utilisation to obtain optimal solutions. FSM simplifies the complex application into simple classified search steps in the overall model to help and give developers a good visualisation of the model. In order to declare each state of our approach, we therefore, examine our model's design with a Jflap tool using two main tests: step test-by-state and multiple run tests. These tests showed the validation of the model by declaring a green sign with each state with typical results. Moreover, we inspect these states by testing most possibilities to achieve expected outcomes via using test methods to obtain optimal results.

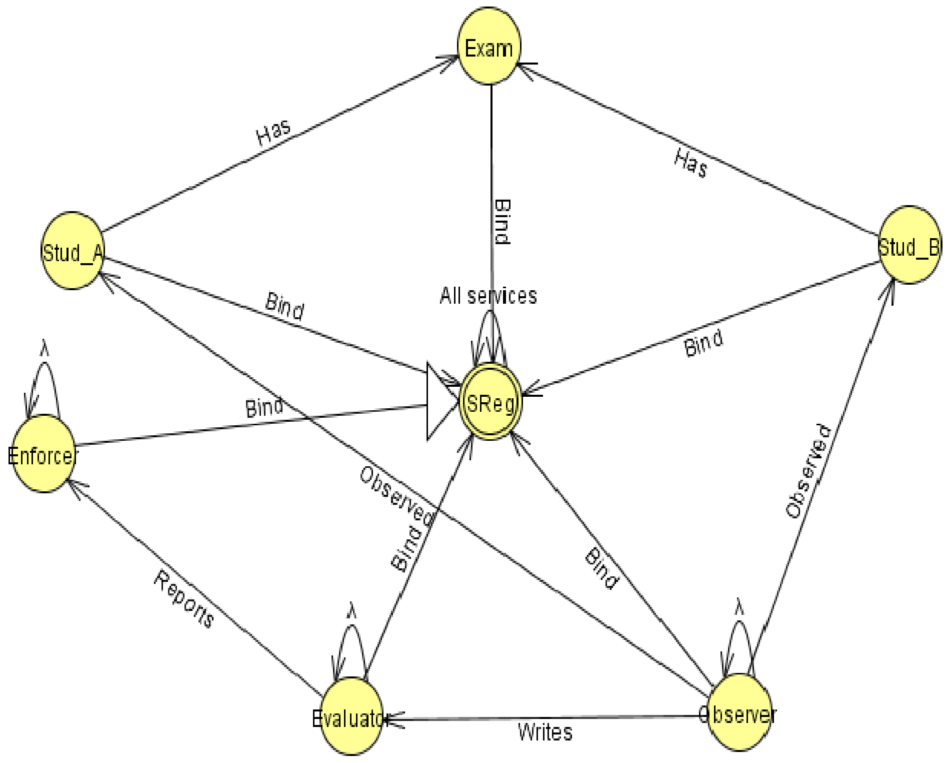


Figure 8.7 The Observation diagram by an FSM and JFLAP to validate proposed approach

```
private class InputFormObserver implements Observer {
      public void update(Observable ob, Object o) {
         doSomeUpdate();
         if (obsInput.countObservers()>0)
               obsInput.deleteObservers();
         obsInput = inputForm.getInputInfo();
         obsInput.addObserver(input);
      }
 }
```

Figure 8.8, Implementing the observer methods

However, in figure 8.9, the validation of the proposed approach has also been examined by the AGG tool to prove and demonstrate the practical applicability of the prototype system by complying with the utilisation of information technologies. AGG uses a specification technique for a set of particular kinds of systems, especially in situations where states exist as complex structures that can be adequately modelled as graphs and in which the behaviour involves a large amount of parallelism. AGG can be described as reactions to stimuli that can be observed in the state of the system. AGG is a formal language suitable for a set of specifications for a type of computational systems. AGG is a development environment tool for attributed graph transformation systems supported by an algebraic approach.

In figure 8.9 below, as elaborated in the previous chapters, the scenario is shown that, all these components are under a confederation system so, the student-a, student-b (Service 1 & 2 or client 1 & 2) and exam have a pre-agreement to be observed. Student-a & student-b have an exam and they are under observation by the observer which should observe the interaction communication between them. In fact the observer observes (partial) behaviours of these services and complete behaviours that can be analysed and compared with security requirements. As a result, we have two sets of behaviours: (a) the observed (partial) behaviours; and (b) the actual required behaviours. Given these two sets can be then compared, it can be determined if the system satisfies the security requirements. If the two sets are the same then the system satisfies the requirements; otherwise if it does not the observer will report to the evaluator and then to the enforcer.

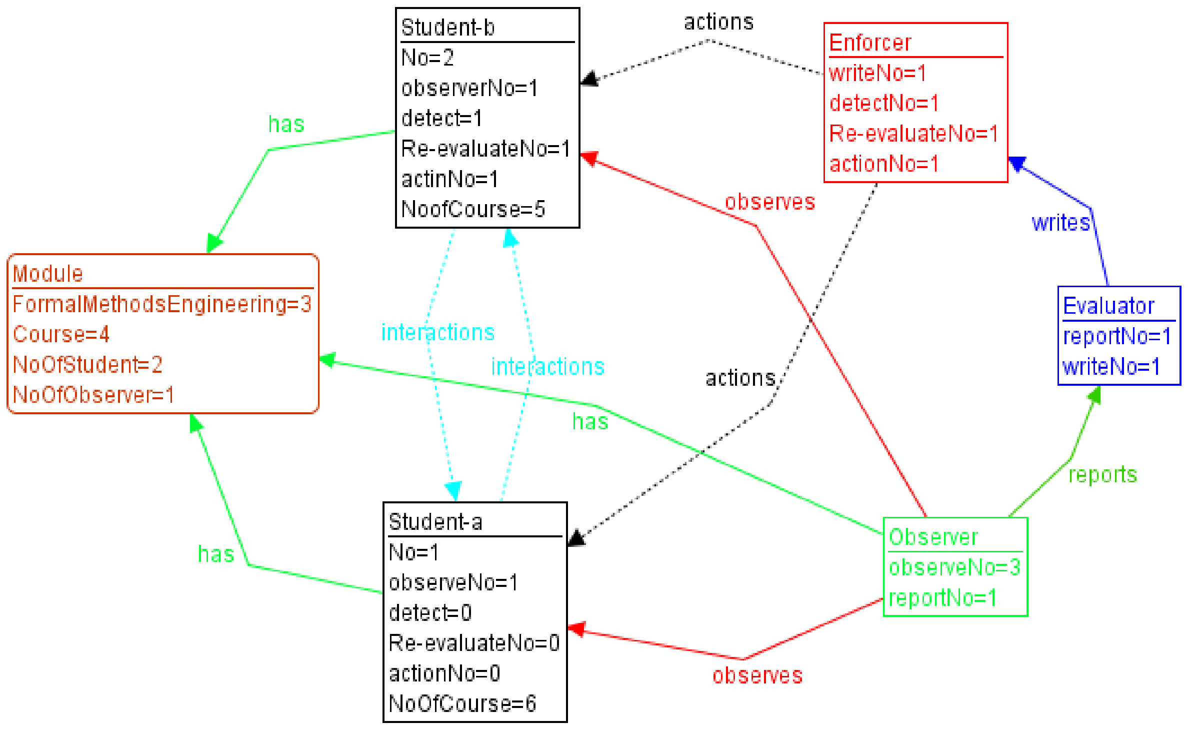


Figure 8.9 The Observation diagram by AGG tool to validate proposed approach

8.2.2.4 The Algorithm for the observation approach

As mentioned in chapter 7, pseudo code is a process of compact that shows a high-level description of a computer programming algorithm. In algorithm 8.1, the observation method() is described, how services can observed, evaluated and then enforced. This algorithm has shown the compact of implementing the main method of the proposed technique which is impacted on the observation system.

Moreover, this algorithm typically omits the details that are not important for understanding the algorithm, such as variable declarations. The purpose of using pseudo code is that it gives the ability to understand it more easily than conventional programming language code. It is also sketching out the structure of the programme before the actual coding takes place.

Algorithm of observation method()

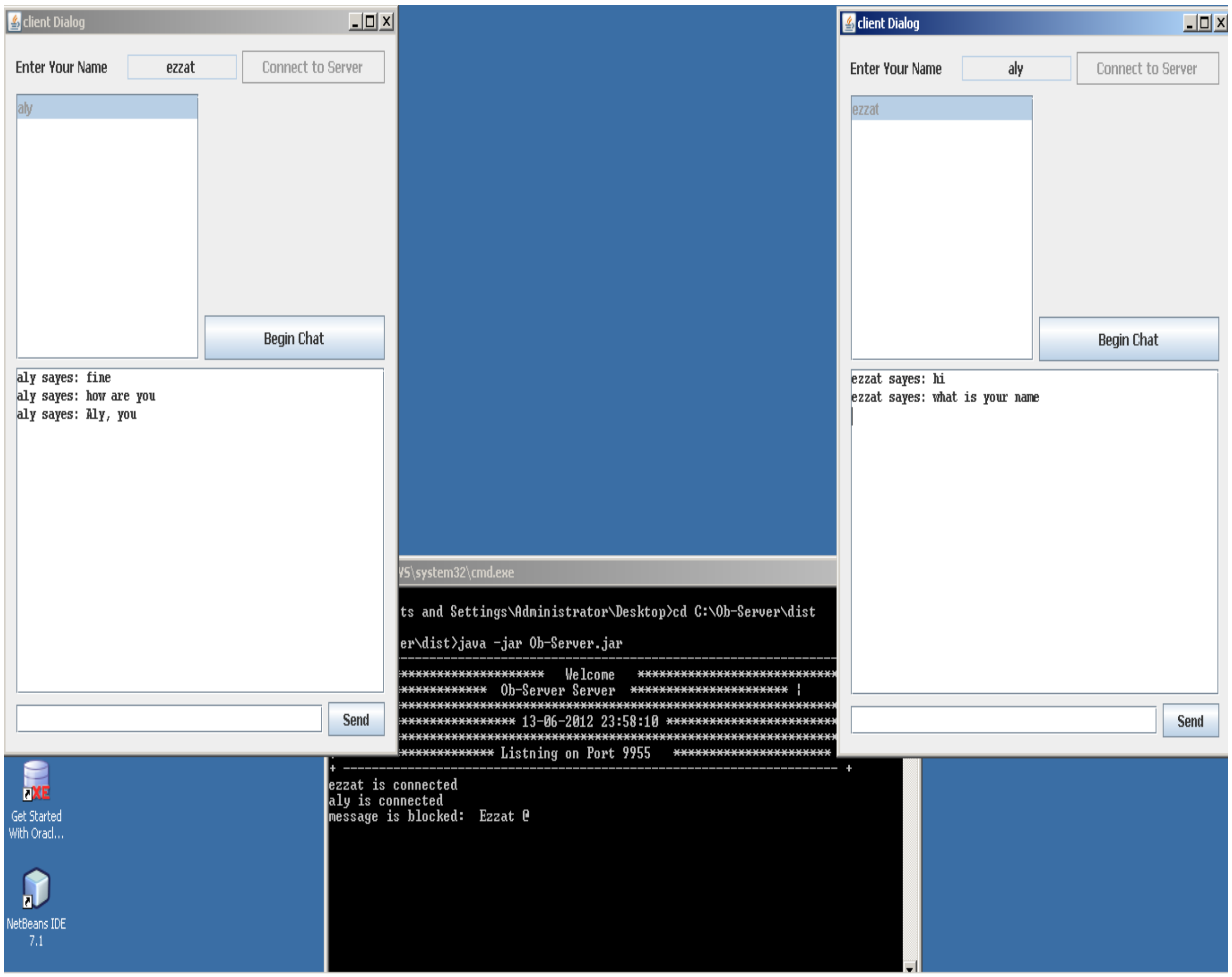
```
Variables=counter, Interact_of_services[A], List_of_services[L], Return_of_services[R], Report[P]
Begin
                For all services ∈ All_Services
                  All_Services←getAll_Services
do
          do    Counter ←getServices_no                                                  (i)
                List_of_services[L]←All_Services
                   For i=0; i < counter ; i++
                      Read(list_of_services[L])
                   End for

      // observer method ()
          do    Interact_of_services[A]← getAll_Services                                 (ii)
                   For i=0; i< counter ; i++
                      getRotate Interact_of_services[A]
                        Read(Interact_of_services[A])
                If anObservable is registered with observer, add anObservable to the list of observers
                   Otherwise, register anObservable and add anObserver to its list of observers.
                   If found interaction between any services ∈ All_services
                   then   Insert Interact_of_services[A]  into Return_of_services[R]
                             .....
                   else if   no interaction found with ∈ All_services == ∞
                                      ......
                End for

      // evaluator method ()
          do    While Return_of_services[R] ≠ Null                                       (iii)
                   For i=0; i < counter;  i++
                      Read(Return_of_services [R])
                        Evaluate interaction into Return_of_services
                        If  (Return_of_services [R]) ≠ satisfy
                        else if   (Return_of_services [R]) ='deviation'
                           then   Report[P]← getReturn_of_services [R]
                   End for
                End while

      // enforcer method()                                                               (iv)
          do    For i=0; i < counter; i++
                   check(Report[P])
                   if (Report[P]='deviation')
                      Detect 'deviation' in (report[P])
                      then   Re-evaluate (report[P])
                      else if   Still has'deviation'
                                Report[P] is == 'deviation'
                         then   Report[P] ≠  satisfy
                                get.remove('deviation')
                end for
End
```

Algorithm 8.1: the main observation method ()

8.2.2.5 Implementing the Observation system by Sun Java

As in figure 8.10, we have used Sun Java to implement and run the observer technique with its services or clients to prove the feasibility of the planned approach. So, we have assumed two clients that deal with each other and are under observation at the same time. As we assumed in the confederation system, these clients interact by messaging. However, to detect any deviation we have assumed that any words which include symbols e.g. ($,%,@, ? and #, it can be added to any word or symbol) will be defined as a deviation or violation.

In figure 8.10 below, to run the program we have firstly to *re-name* each client and then press on *connect-to-server* to link the client to be observed. Afterwards, the *Observer* will check all interaction messages between clients and if any messages have unaccepted symbols then they will be detected by adding it to the *blockSymbols file*.



```
aly sayes: fine
aly sayes: how are you
aly sayes: Aly, you
```

```
ezzat sayes: hi
ezzat sayes: what is your name
```

```
ts and Settings\Administrator\Desktop>cd C:\Ob-Server\dist

er\dist>java -jar Ob-Server.jar
------------------------------------------------------------
********************   Welcome   **************************
************  Ob-Server Server  ********************** ¦
***********************************************************
**************** 13-06-2012 23:58:10 **********************
***********************************************************
************* Listning on Port 9955   *********************
+ ---------------------------------------------------------- +
ezzat is connected
aly is connected
message is blocked:  Ezzat @
```

Fig 8.10 Snapshot of running and testing the observation

8.2.2.6 Evaluation

This program is done by Sun Java to present the e-observer technique. The e-observer was explained and elaborated within chapter four. The usage of observation has rapidly increased in recent years and a number of enterprises have required it.

In figure 8.10 above, the program has shown that the e-observer technique is an important aspect for many action research studies that are used for many purposes. The use of the e-observer technique eases the attachment and detachment of monitoring agents to software probes and reduces the information distribution complexity. Further-more, monitoring agents can again become observable objects, allowing other agents to subscribe to their processed information. This mechanism is the foundation for building complex multi agent networks inside the agent concentrator component for monitoring the protocol stack instances.

8.3 Summary

This chapter has analysed and evaluated the eStudent system prototype, which is embedded within the observation system. The observation system includes important modules, which have the ability to keep the system safe.

At the beginning of this chapter, we considered the value of the eStudent as a proposed system to prove the observation approach. The security requirements of the observation system, based on Web services, have also been considered during this chapter. Moreover, the validation of the observation system has been shown by using FSM, JFLAP and AGG tools which have demonstrated the feasibility of the proposed approach and illustrated the methodology that could produce a high performance. The main issue in this chapter is to prove the validation of the e-observer approach which has been done and provided by Sun Java. Java is an object-programming language that is used for its facility and usage. We have used Java to design and implement this technique to demonstrate the applicability of this approach.

The evaluation that we used has taken into consideration several tools to analyse the outcome of the results to validate and obtain satisfactory results of the system.

The next chapter will conclude and summarise the work presented by this thesis. The significance of the main findings of the research will be presented. Further, it will highlight the most important contributions that have been made. The chapter will then conclude with a discussion of methods and direction for possible future studies in this field.

CHAPTER 9

Conclusion and Future Research

Objectives

- *Summarising our findings, achievements and suggestions for future research.*

9.1 Introduction

This analysis proposes the principles, rules and postulates that were contained in this research. The analysis has been shaped by the integration of three main, important fields in IT: the SOC, observation technique and service behaviour. All these fields come to play increasingly important roles in the development of a strategy of security. This means that there is a growing demand for the methodologies and technologies that support these fields for their different purposes. The most important challenge for this investigation was to find a suitable technology that develops an approach to ensure security requirements are satisfied and enforced.

This thesis focused upon examining the adoption and dissemination of security policy in a Web service environment. In addition, it traced the roots of Web services from its background and related work in the architectural built environment to the present. Conflicting views of its associated parts were critically analysed for an optimal solution. Afterwards, the thesis presented a critical assessment of the observation technique in terms of a structure for documenting information. This research was prompted by the scarcity of resources for organisations that wish to introduce security into their systems.

Moreover, this study discussed a leverage of security policy implementation in web service environments and its associated part. This should help to discover a notion technique through policy-based QoS to find the desired information and solution with additional features for policy.

9.2 Summary

This thesis has addressed the key issues which manage and enable tracing services during the process of communication in the Web environment. We proposed the observation system that provides a meaningful surveillance to minimise and control security leaks. This system can be used in many purposes by different organisations. We dedicated most of our research to present a comprehensive technique to develop an infrastructure for the emerging concept of

Web services. The key in this technique relates to security policy architecture based on Web services.

In addition, the thesis has described the key points for an SOC, which involves extended loosely coupled activities among several systems. Web services, as a part of the SOC, are becoming an important technology in the evolution of distributed computing and Web throughout the world. Web services are dynamic and independent entities offering a variety of miscellaneous functionalities on the Web. For a given functionality, several Web services may compete with their offerings.

The main aim of this study is to tame an appropriate solution so that it is able to adopt the observation system, which is based on Web services and their related technologies. This approach has provided a rich context for a set of robust core capabilities that enable observing services to foster an efficient collaboration. The outcome results were then evaluated and subsequently detected and enforced by the enforcer model. The enforcer technique is a fundamental step towards attaining the envisioned controlling of the resources.

In our research, we formulated compositional rules that resolved, by combining and verifying these rules, to overcome any violation activities in the system. A major purpose of these rules is to express the requirements at a high level of abstraction, hiding the details of the implementation that is necessary for their enforcement. To efficiently deploy such a scheme, we proposed design of optimisation strategy for the enforcer model, which selects an optimal way for its successful detection deployment over Web services.

The desire of this thesis was to explain and analyse the interaction communication process between services under the Observation system based on the significance of Web environment. By having this view in mind, the focus throughout the thesis was the field of Web services, Observation systems and behaviour to identify the affecting factors.

9.3 Research Question Revisited

To evaluate the research presented in this thesis, which emphasises the significance of the contributions, the research questions are re-visited. The overall research question investigated and presented in Chapter one was:

- ✓ *How to build a secure system from vulnerable components? Those components may be (web) services?*

Fundamental to this question is to study the service behaviour. So construction of the behaviour of a system can only be achieved by observing the communication between services, via demonstrating the process of building the practical applicability with the utilisation of information technologies. Also, a rigorous technique for determining appropriate solutions has been considered and presented.

The questions were addressed in general terms by proposing an observation framework and the novelty of the proposed framework derives from a consideration of three main aspects: SOC, observation process itself and behaviour.

9.4 Thesis Contributions

The major focus of this thesis is to ensure that security requirements are satisfied by investigating and studying the behaviour of services during communication. In addition, to ascertain how effectively services can be contacted and implemented. Due to the sheer size of the services that are provided by many vendors, to achieve our goals, we looked at different issues and made several contributions, these contributions constitute the underlying infrastructure upon which to build a comprehensive infrastructure to support Web services.

The main contribution towards this work is to form a rigorous approach to specify and verify the behaviour of services with the aim of simplifying the task of designing and implementing the Observation system and their interaction requirements. Hence the main contributions of the research are:

- The proposal and development of the Observation system that increases surveillance by observing interactive communication between services and then sends outcome results to the evaluator model after processing them.
- The taming of the design complexity of the Observation Model by leaving considerable degrees of freedom for their structure and behaviour and by bestowing upon them certain characteristics and to learn and adapt with respect to dynamically changing environments.
- Formulate novel policy-based techniques, which support the Observation Model in monitoring the services that interact with each other by verifying them in a formal and systematic manner. We developed a set of rules using AGG tools, which can be applied inductively to verify sets of traces that are generated by a specification of the policy. These rules could also be used for verifying the functionality of a system.
- Design enforcement architecture that uses a technique to detect any violation activity by addressing the security policy system using a systematic approach that can be leveraged by the model's requirements within the Web context,
- Perform the enforcement system for successfully deploying infrastructure as a Web service to define the optimisation model that would capture efficiency requirements via addressing this enforcement system and a systematic approach that can be leveraged by the model's specifications.
- Implement a prototype system that uses the observation technique to control, manage and reduce risk by adopting those protection features of the enforcement system that can specify a method that concisely describes the set of traces generated by the enforcement tool.

As well as these contributions, other important contributions relate to the knowledge that is embedded within this thesis and based on the above contributions, such as:

- Propose and design an Observation Model, which consists of an enhanced observer, evaluator and enforcer models, which works via an e-observer that will monitor the behaviour of service and then report them to the evaluator model, which will evaluate the outcomes and then send them to the enforcer model to be detected.

- A technique of the e-observer that uses a proactive or precaution system to minimise the risk to the resources.
- The enforcer model uses an independent technique, which means it can be used with any other solution owing to its dynamicity.

The contents of the thesis are précised chapter by chapter by the following summary.

In chapter 2, we conducted a comprehensive review of SOC, policy based approaches and QoS, which formed a framework to discuss and comment upon the relevant literature, including an overview of the literature on Web services, and their scenarios, benefits and architecture. In addition, it describes the Web Services Protocol Stack in detail with XML, WSDL and UDDI languages.

In chapter 3, we provided, in depth, an overview of the background information that influences security policies. This background information includes a definition of security policy, access control models, security policy languages, security threats and security goals. The chapter also discussed the impact of security models.

Chapter 4, we proposed and designed an Observation Architecture, which managed the surveillance technique to reduce risk. This chapter elaborated on the Observation and its associated parts.

In chapter 5, we formulated a policy-based technique for the verification of the observation approach. This chapter described policy specification languages e.g. ponder policy. In addition, it provided and made rules designed by AGG tools for simulation and verification.

In chapter 6, we designed and developed the enforcement architecture that provided a technique for detecting violations, as well as to minimise the risk to resources by using a proactive approach.

In chapter 7, a prototype system was developed that focused upon the experimental conduct and implementation. The Observation system was embedded in this prototype to manage security.

In chapter 8, we provided an evaluation of the prototype system to demonstrate the practical applicability of the proposed approach by using tools e.g. AGG and FSM with JFLAP to prove and validate the feasibility of this prototype which integrated with the proposed technique.

Chapter 11 summarises the research presented in this thesis, highlights the significance of the proposed contributions and discusses directions for possible future work in this field.

9.5 Future Research

This research is ongoing. We feel that further study is required in order to analyse some of the main features of this work, namely, the SOC and observation approach based upon the security policy in order to penetrate more deeply to discover and obtain the visibility, facility, power and performance of these fields.

Several extensions to the observation infrastructure can be made. One of them is to enhance the developed evaluator model to optimise techniques for Web services. This would require using intelligent systems that take advantage of the current context. Evaluation as a technique is a significant issue as it is related to many sciences, such as, mathematics and computing. Another extension is to cater for the dynamic and volatile nature of Web services. An adaptive approach needs to be designed to ameliorate the effects upon the efficiency of the service's execution plan when unpredictable events occur during run time. Another extension is to the main part of the enforcer model, where the Policy enforcer (PE) should address the following:

- The dynamic policy priority should increase for prompt response in case of any violation activity.
- The multi-domain should handle and control multiple attacks simultaneously.

Bibliography

1. Al-Ajlan, A. (2008). Service Oriented Computing for Dynamic Virtual Learning Environments (Moodle). STRL. Leicester,UK, De Montfort. PhD Thesis: 328.

2. Aldrawiesh, K., F. Siewe, et al. (2011). An observation model to detect security violations in web services environment. Proceedings of the 2011 International Conference on Intelligent Semantic Web-Services and Applications. Amman, Jordan, ACM: 1-6.

3. Alferes, J. J., F. Banti, et al. (2006). An event-condition-action logic programming language. Proceedings of the 10th European conference on Logics in Artificial Intelligence. Liverpool, UK, Springer-Verlag: 29-42.

4. Alter, M. (2005). Web Services.

5. Andrew, C., C. Geoff, et al. (1994). "A quality of service architecture." SIGCOMM Comput. Commun. Rev. 24(2): 6-27.

6. Andrzej Uszok, Jeffrey M. Bradshaw, et al. (2004). "Policy and Contract Management for Semantic Web Services ".

7. Avik, S. and P. Amit (2006). Model-based functional conformance testing of web services operating on persistent data. Proceedings of the 2006 workshop on Testing, analysis, and verification of web services and applications. Portland, Maine, ACM.

8. Azadeh, M. A., M. Anvari, et al. (2007). An integrated FDEA-PCA method as decision making model and computer simulation for system optimization. Proceedings of the 2007 summer computer simulation conference. San Diego, California, Society for Computer Simulation International: 609-616.

9. Bishop, M. (2002). The Art and Science of Computer Security, Addison-Wesley Longman Publishing Co., Inc.

10. Boxman, J. (2005). A Practical Guide to Linux Traffic Control.

11. Bussler, D. F. a. C. (2002). The Web Service Modeling Framework WSMF. De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands.

12. C. Matthew MacKenzie, Ken Laskey, et al. (2006). Reference Model for Service Oriented Architecture OASIS Standard.

13. Cavanaugh, E. (2006). Web services: Benefits, challenges, and a unique, visual development solution, Product Marketing Manager, Altova® WhitePaper.

14. CÖMERT, C. (2004). Web Services and National Spatial Data Infrastructure (NSDI). Trabzon, Turkey, KTU, 6108

15. Corp, N. n. (2003). Introduction Quality of Service (QoS) Nortel networks Corp, White Paper.

16. Coyle, F. P. (2002). XML, WEB SERVICES, AND THE DATA REVOLUTION, Addison-Wesley

17. D. Austin, A. Barbir, et al. (2004). "Web Services Architecture Requirements." from http://www.w3.org/TR/wsa-reqs/.

18. Damianou, N. Dulay, et al. (2001). Ponder: A language for specifying security and management policies for distributed systems, the language specification, , Imperial College of Science Technology and Medicine, Department of Computing, London, UK. 2.3,.

19. Damianou, N., N. Dulay, et al. (2002). Tools for Domain-based Policy Management of Distributed Systems. NOMS, IEEE/IFIP, Network Operations and Management Symposium, 2002. , London, UK, IEEE.

20. David Booth, Hugo Haas, et al. (2004). Web Services Architecture, World Wide Web Consortium(W3C).

21. David Sprott and L. Wilkes (2004). "Understanding Service-Oriented Architecture." Microsoft Architect Journal.

22. Davies, N. J., D. Fensel, et al. (2004). "The Future of Web Services." BT Technology Journal 22(1): 118-130.

23. Denis, V. (2006). "Security Policies and the Software Developer." IEEE Security and Privacy 4(4): 42-49.

24. E. Lupu (1998). A Role-Based Framework for Distributed Systems Management. Department of Computing London UK, Imperial College. PhD Thesis.

25. Eales, A. (2005). The Observer Pattern Revisited. Educating, Innovating & Transforming: Educators in IT Concise paper,NACCQ05.

26. Edgardo, A., B. Marco, et al. (2006). When is it convenient to predict the web services completion time? Proceedings of the 24th IASTED international conference on Parallel and distributed computing and networks. Innsbruck, Austria, ACTA Press.

27. Edward, G. A. (1994). Fundamentals of computer security technology, Prentice-Hall, Inc.

28. Ethan, C. (2002). Web Services Essentials, O'Reilly \&amp; Associates, Inc.

29. Evan, M. (2007). Testing and Analysis of Access Control Policies. Companion to the proceedings of the 29th International Conference on Software Engineering, IEEE Computer Society.

30. F. García, G. Martínez, et al. (2005). Representing Security Policies in Web Information Systems. WWW 2005. Chiba, Japan.

31. Firesmith, D. G. (2003). "Security Use Cases " Journal of Object Technology 2(3): 12.

32. Foster, I. and C. K. (1998). The Grid:Blueprint for a New Computing Infrastructure. San Francisco, California, Morgan Kaufmann.

33. Francisco, C., D. Matthew, et al. (2002). "Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI." IEEE Internet Computing 6(2): 86-93.

34. Frank, P. C. (2002). Xml, Web Services, and the Data Revolution, Addison-Wesley Longman Publishing Co., Inc.

35. Fred, B. S. (2000). "Enforceable security policies." ACM Trans. Inf. Syst. Secur. 3(1): 30-50.

36. Goethals, F. (2002). 4 New directions in Application Integration: Web Services: 21-44.

37. Gorton, S. and S. Reiff-Marganiec (2007). Policy-driven Business Management over Web Services Integrated Network Management, IFIP/IEEE International Symposium

38. H. Janicke (2007). The Development of Secure Multi-Agent Systems. Software Technology Research Laboratory. Leicester,UK, De Montfort Ph.D Thesis: 219.

39. H. Janicke, F. Siewe, et al. (2006). Analysis and Run-time Verification of Dynamic Security Policies Defence Applications of Multi-Agent Systems Springer Berlin / Heidelberg.

40. Hans Weigand, Paul Johannesson, et al. (2008). Value-based Service Design Based On A General Service Architecture. IFIP International Federation for Information Processing BUSITAL'08.

41. Harel, D. and M. Polit (1998). Modeling Reactive Systems with Statecharts: The Statemate Approach, Mcgraw-Hill (Tx).

42. Harold F. Tipton and M. Krause (2009). Information Security Management Handbook, Sixth Edition, Auerbach Publications.

43. Harris, S. (2008). CISSP Certification All-in-One Exam Guide, Fourth Edition, McGraw-Hill, Inc.

44. Hedi Hamdi, Adel Bouhoula, et al. (2007). A Software Architecture for Automatic Security Policy Enforcement in Distributed Systems. Proceedings of the The International Conference on Emerging Security Information, Systems, and Technologies, IEEE Computer Society.

45. Hugo Haas and A. Brown (2004). Web Services Glossary, World Wide Web Consortium(W3C).

46. Huhns, M. N. and a. M. P. Singh (Feb 2005). Service-Oriented Computing: Semantics, Processes, Agents. England, Johe Wiley & Sons ltd.

47. Jagadeesh, Nandigam, et al. (2006). "A tool for experimenting with web services." J. Comput. Small Coll. 22(1): 36-45.

48. Jeffrey, G., J. Marijn, et al. (2004). The advantages of web service orchestration in perspective. Proceedings of the 6th international conference on Electronic commerce. Delft, The Netherlands, ACM.

49. Jian, Y. (2003). "Web service componentization." Commun. ACM 46(10): 35-40.

50. Jianchun, F. and K. Subbarao (2005). "A snapshot of public web services." SIGMOD Rec. 34(1): 24-32.

51. Jürgen Branke, M. C. Moez, et al. (2006). Organic Computing - Addressing Complexity by Controlled Self-Organization. Proceedings of the Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation, IEEE Computer Society.

52. K. Aldrawiesh, A. Al-Ajlan, et al. (2009). "A Comparative Study between Computer Programming Languages for Developing Distributed Systems in Web Environment." ICCIT2009, ACM.

53. K. Aldrawiesh, F. Siewe, et al. (2011). An Observation Model to Detect Security Violations in Web Services Environment. The International conference on Intelligent Semantic Web-Services and Applications (ISWSA 2011), Amman, Jordan, Isra University (ISWSA 2011).

54. K. Aldrawiesh, A. Platt, et al. (2011). Towards Development a Policy-Based Technique for Enforcing Security Violations. The 5th Saudi International Conference, in proceedings of ICT Coventry, UK, SIC2011.

55. K. Aldrawiesh, F. Siewe, et al. (2011). An observation model to detect security violations in web services environment. Proceedings of the 2011 International Conference on Intelligent Semantic Web-Services and Applications. Amman, Jordan, ACM: 1-6.

56. K. Gottschalk, S. Graham, et al. (2002). "Introduction to Web Services Architecture." IBM Systems journal 41(2).

57. Karila, A. (1991). Open Systems Security – an Architectural Framework. Helsinki FINLAND, Helsinki University of Technology. PhD: 143.

58. Katia Sycara, M. Paolucci, et al. (2005). Combining Services and Semantics on the Web. Cambridge, MA.

59. M. Papazoglou1, P. Traverso, et al. (2006). Service-Oriented Computing Research Roadmap. Dagstuhl Seminar Proceedings 05462 Service Oriented Computing (SOC).

60. M. Papazoglou and D. Georgakopoulos (2003). "Introduction to Service-oriented computing (SOC)." Commun. ACM 46(10): 24-28.

61. M. Papazoglou, P. Traverso, et al. (2006). Service-Oriented Computing Research Roadmap. Dagstuhl Seminar Proceedings 05462 Service Oriented Computing (SOC).

62. M.Huhns and M. S. (2005). "Service-Oriented Computing- Key Concepts and Principle." IEEE Internet Computing.

63. Mani, A. and A. Nagarajan (2002). Understanding quality of service for Web services, IBM developerWorks.

64. Mario, M. (2007). QoS Over Heterogeneous Networks, Wiley Publishing.

65. Massimo, M., O. Mourad, et al. (2006). Access control enforcement for conversation-based web services. Proceedings of the 15th international conference on World Wide Web. Edinburgh, Scotland, ACM.

66. Michelson, B. M. (2008). Service Discovery Using Customer Scenario® Mapping, Enterpriseleadership.org

67. Myerson, J. (2002). "Advancing the Web services stacks." from http://www.ibm.com/developerworks/webservices/library/ws-wsa/#main.

68. N. Damianou, N. Dulay, et al. (2000). Ponder: A Language for Specifying Security and Management Policies for Distributed Systems. London, Imperial College of Science Technology and Medicine, Department of Computing.

69. N. Damianou, N. Dulay, et al. (2001). The Ponder Policy Specification Language. Workshop on Policies for Distributed Systems and Networks, Bristol, Springer

70. Nabor, C. M., Jos, et al. (2005). An empirical evaluation of client-side server selection policies for accessing replicated web services. Proceedings of the 2005 ACM symposium on Applied computing. Santa Fe, New Mexico, ACM.

71. Newcomer, E. (2002). Understanding Web Services, XML/ WSOL/ SOA and UDDI, Addison-Wesley Professional.

72. Nirmal, K. M., K. Ravi, et al. (2004). Cooperative middleware specialization for service oriented architectures. Proceedings of the 13th international World Wide Web conference on Alternate track papers \&amp; posters. New York, NY, USA, ACM.

73. Oriyano, S.-P. (2008). Why a security policy? Introductory, IBM Corporation: 10.

74. Panagiotis, L. (2006). "SOAP and Web Services." IEEE Softw. 23(6): 62-67.

75. Papazoglou, M. P. (2003). Service-oriented computing: concepts, characteristics and directions. Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference

76. Parisi-Presicce, F. and Y. Zhao (2005). "Policy Analysis and Verification by Graph Transformation Tools." Electron. Notes Theor. Comput. Sci. 127(1): 101-112.

77. Paul Ferguson and G. Huston (1998). Quality of Service: Delivering QoS on the Internet and in Corporate Networks, John Wiley & Sons.

78. Peri, R. V. (1996). Specification and Verification of Security Policies. Faculty of the School of Engineering and Applied Science. Virginia, University of Virginia. Ph.D: 179.

79. Petrone, G., L. Ardissono, et al. (2003). Enabling conversations with web services. Proceedings of the second international joint conference on Autonomous agents and multiagent systems. Melbourne, Australia, ACM.

80. Rajagopalan, S. R. (2004). Automatic Security Policy Management in Modern Networks Guarding Your Business, Springer US.

81. Raju Rajan, Dinesh Verma, et al. (1999). A policy framework for integrated and differentiated services in the Internet. IEEE Communications Society IEEE.

82. Ravi, D. and T. Pallavi (2006). "Web services demystified." J. Comput. Small Coll. 21(5): 1-2.

83. Richards, R. (2006). Pro PHP XML and Web Services, Apress.

84. Roberts, J. (2002). Functional Requirements for Describing Services Discussion paper for DC-Gov. New Zealand.

85. S. Parsa and M. Damanafshan (2007). Seamless Secure Development of Systems: From Modeling to Enforcement of Access Control Policies. AICCSA '07. IEEE/ACS International Conference on Computer Systems and Applications, 2007., Amman, IEEE.

86. S. Weerawarana, F. Curbera, et al. (2006). Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging Prentice Hall

87. Sattarova Feruza, Y. (2008). Advanced Security Policy Implementation for Information Systems. Proceedings of the 2008 International Symposium on Ubiquitous Multimedia Computing, IEEE Computer Society.

88. Schmitter, E. (1996). Modelling, Analysis and Evaluation of Systems Architectures. Selected papers from the 4th International Workshop on Computer Aided System Theory, Springer-Verlag: 241-251.

89. Shalom, T., A. Serge, et al. (2001). Are Web Services the Next Revolution in e-Commerce? (Panel). Proceedings of the 27th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc.

90. Shimonski, R. J. (2003) Defining a Security Policy. Articles / Misc Network Security 2,

91. Shirey, R. (2000). Internet Security Glossary, RFC Editor.

92. Siewe, F. o. (2005). A Compositional Framework for the Development of Secure Access Control Systems. STRL. Leicester,UK, De Mont Fort. Ph.D Thesis: 238.

93. Sleeper, B., and, et al. (2001). "Defining Web Services." from http://www.site.uottawa.ca/~stan/csi5389/readings/wsdefined.pdfhttp://www.site.uottawa.ca/~stan/csi5389/readings/wsdefined.pdf.

94. Sloman, M. (1994). "POLICY DRIVEN MANAGEMENT FOR DISTRIBUTED SYSTEMS " Journal of Network and Systems Management 2: 22.

95. Stefan, T., K. Rania, et al. (2004). Composition of coordinated web services. Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware. Toronto, Canada, Springer-Verlag New York, Inc.

96. Stephen, J. H. Y., S. F. H. James, et al. (2005). Composition and evaluation of trustworthy Web Services. Proceedings of the IEEE EEE05 international workshop on Business services networks. Hong Kong, IEEE Press.

97. Steve, V. (2003). "Service Discovery 101." IEEE Internet Computing 7(1): 69-71.

98. Syed Naqvi and Philippe Massonet (2006). A Study of Languages for the Specification of Grid Security Policies, CoreGRID Technical Report TR-0037.

99. Symon, C., C. Qiming, et al. (2003). Managing Security Policy in a Large Distributed Web Services Environment. Proceedings of the 27th Annual International Conference on Computer Software and Applications, IEEE Computer Society.

100. Systems CISCO, I. (2001). Internetworking Technologies Handbook (Cisco Core) Cisco Press

101. Taentzer, G. (1997). "The Attributed Graph Grammar System, AGG Site, from http://user.cs.tu-berlin.de/~gragra/agg/index.html.

102. Tian, M., T. V. , et al. (2003). "Performance Considerations for Mobile Web Services."

103. Tsai, W. T. and Y. C. (2006). Introduction to Service-Oriented Computing. Arizona, USA.

104. Utkarsh, S., M. Kamesh, et al. (2006). Query optimization over web services. Proceedings of the 32nd international conference on Very large data bases. Seoul, Korea, VLDB Endowment.

105. V.. Hu, E. Martin, et al. (2007). Conformance Checking of Access Control Policies Specified in XACML. Proceedings of the 31st Annual International Computer Software and Applications Conference - Vol. 2- (COMPSAC 2007) - Volume 02, Beijing, China IEEE Computer Society.

106. Vasudevan, V. (2001). "Web services Primer." from <http://webservices.xml.com/pub/a/ws/2001/04/04/webservices/index.html>.

107. Verma, D. (2000). Policy-Based Networking: Architecture and Algorithms, Sams.

108. Vincent C. Hu, Evan Martin, et al. (2007). Conformance Checking of Access Control Policies Specified in XACML. 31st Annual International Computer Software and Applications Conference(COMPSAC 2007), IEEE Computer Society.

109. Vincent, T. P. (2002). Future of web services, Cognizant Technology Solutions 9.

110. Xiaoyuan, T. (2007). Internet Quality of Service Monitoring System, VDM Verlag.

111. Xie, T. (2008). Testing and Verification of Security Policies. Automated Software Engineering Research Group North Carolina State, North Carolina State University.

112. Y. Zhao and F. Parisi-Presicce (2004). Policy Analysis and Verification by Graph Transformation Tools. Electronic Notes in Theoretical Computer Science, USA, elsevier.