

A web-based policy editor for SANTA	العنوان:
Al Mutairi, Abd Algder	المؤلف الرئيسي:
Siewe, Francois(Supervisor)	مؤلفين آخرين:
2009	التاريخ الميلادي:
Leicester	موقع:
1 - 105	الصفحات:
687206	رقم MD:
رسائل جامعية	نوع المحتوى:
English	اللغة:
رسالة ماجستير	الدرجة العلمية:
De Montfort University	الجامعة:
Faculty of Computing Sciences and Engineering	الكلية:
بريطانيا	الدولة:
Dissertations	قواعد المعلومات:
الأمن المعلوماتي، هندسة البرمجيات، الإنترنت، لغة البرامج	مواضيع:
https://search.mandumah.com/Record/687206	رابط:

Chapter 1: Introduction

1-1- Motivation

The expansion of the internet and networks in the world led to increase the number of security threats and breaches. Therefore, security should be constructed in the early stage for system development [11]. Hence, the developers are focusing on security requirements and how to achieve three key aspects or requirements: confidentiality, integrity and availability.

The major purpose of the security requirements is the prevention of the data resources from threats. This is achieved by setting up constraints that define individuals permitted access to the data assets of a system. However, confidentiality requires that only authorized individuals are allowed to access computer system resources, whereas integrity describes the prevention of an unofficial modification of data. Availability of required resource or information is its ability to be used [1].

Security policy is applied as a basis of configuration and auditing of computer systems and networks, while upholding obedience with the policy requirements. These allow later implementation of developmental, operational guidelines and introduction of user access monitoring regulations [6].

Security is a major issue for organisation which deals directly with internet by doing online business or providing services. The leak of security can facilitate the increase exposure of the threat to the internet which can cause loss or damage of the information or equipment or threat human life in some critical system.

This project encourages me to solve the problem that result from the difficulty of using SANTA language for non specialist user. It will deliver a friendly editor that can allow the user to create his policy easily. This policy will be created by using SANTA structure.

1-2- Original Contribution:

In this dissertation, I have designed and implemented a user-friendly web-based policy editor that allows non specialist users (normal users) to create and manage security policies in SANTA. It is a language for specifying security policies such as authorization and obligation policies. It is developed in the Software Technology Research Laboratory (STRL) at DMU. Policies specified in SANTA can be analyzed using a run-time verification tool called Tempura.

This editor has different interfaces that allow the user to create simple and compound conditions, policy rules, simple policies and compound policies and. In addition, it has interfaces for managing subjects, objects and actions. Furthermore, it also allows the user to conduct functionality upgrades such as modifying, deleting the existing policies and adding new policies. All interfaces connected with database that can manage all policies.

1-3- Objectives

There are many objectives for my project:

- It helps to become skilled at programming languages.
- It helps to understand the concept of the security requirements and security policy.
- It helps to identify the security policy language called SANTA.
- It helps to apply some of my knowledge that learned from the modules.

1-4- Methodology to reach the project

In the analysis stage, the UML (Unified Modeling Language) were used to build the structure of the software. So it helps to identify the functional and non functional requirement. In addition, the modeling of my project was constructed by using JFLAB tools. The ASP.net software with SQL server 2005 was used to implement the project because I found it to be suitable software that has several features. In the later stage Black and white test method were used.

1-5- Project Planning Table

It is important for me to organize my work from the early stage until the final stage. So it helps me to finish the project on time.

ID	Task Name	Start	Finish	Duration
1	Choose Topic and approval by supervisors	25/05/09	1/06/09	1W
2	Literature review	2/06/09	2/07/09	4W
3	Analysis And Design	3/07/09	24/07/09	3W
4	Implementation and testing	25/07/09	3/09/09	6W
5	Writing up and Submission	25/05/09	3/09/09	15W

Table 1: Project Plan

1-6- Resource of the Project

In this project, we used academic paper, books and internet to complete my literature review on this subject area like "Computer Security" for Matt Bishop and supervisor PHD thesis. ASP.net and SQL 2005 software were used to implement the project.

1-7-Tools and Technologies Used

Policy Editor for SANTA is a basic user-friendly system to facilitate the non –specialist users to perform the intended activities. System has been developed on the Microsoft Platform using the following tools and technologies.

- Visual Studio 2008 as development tool.
- Microsoft .Net Framework 3.5 as development framework.
- C# as a language.
- SQL Server Management Studio as Database Tool.
- SQL Server 2005 as Database Server.
- Rational Rose as Designing Tool.

1-8 Dissertation Outline

In Chapter 2, I provide a comprehensive literature review about security requirements and security policy. It provided the definition of a security policy and security requirements. In addition I give an overview of policy models and policy languages. In the final part of the chapter, I present security threats.

In Chapter 3, I identify relevant components of the SANTA policy language. SANTA is a language for specifying security policies such as authorization and obligation policies.

In Chapter 4, I present the Finite State Machine model of the system, designed using JFLAB tools.

In Chapter 5, I analyze and design a user-friendly web-based policy editor. This allows non specialist users (normal users) to create and manage security policies in SANTA.

In Chapter 6, I implement a user-friendly web-based policy editor that allows non specialist users (normal users) to create and manage security policies in SANTA.

In Chapter 7, I test and evaluate the system as whole.

In Chapter 8; the final chapter, I propose conclusions of this dissertation and also provide recommendations for future work on the topic.

A web-based policy editor for SANTA	العنوان:
Al Mutairi, Abd Algder	المؤلف الرئيسي:
Siewe, Francois(Supervisor)	مؤلفين آخرين:
2009	التاريخ الميلادي:
Leicester	موقع:
1 - 105	الصفحات:
687206	رقم MD:
رسائل جامعية	نوع المحتوى:
English	اللغة:
رسالة ماجستير	الدرجة العلمية:
De Montfort University	الجامعة:
Faculty of Computing Sciences and Engineering	الكلية:
بريطانيا	الدولة:
Dissertations	قواعد المعلومات:
الأمن المعلوماتي، هندسة البرمجيات، الإنترنت، لغة البرامج	مواضيع:
https://search.mandumah.com/Record/687206	رابط:

Chapter 2: Literature Review

2-1- Overview

This chapter presents an overview of some significant writings on security requirements and security policy.

2-2- Security Requirements

The main objective of the security requirements are to safe-guard the information data against potential risks. They usually provided as a mean of permitting access to an organization's system resources, whilst setting out a user guidelines for various individuals who have permission to access the system and their limitations of use [1].

2-3- The fundamental Components

The integral components of computer security comprises of confidentiality, integrity and availability. These features are different in their characterization from one another. They also vary in the framework each aspect is applied. However, individual requirements, customs and laws of a specific organization determine the interpretation of the aspect in that particular context or environment [1].

2-3-1- Confidentiality

Confidentiality is defined as "the concealment of information or resources" [1]. The application of computer in sensitive environments (for e.g. government and industry) has necessitated the undisclosed use of information. For instance, revealing the exact statistic of people who have doubts about politicians is not as significance as revealing that the poll itself was carried out by a subordinate of the politicians, hence concealing resources is a feature of confidentiality. Several website hide their application system and configuration as well as organization may conceal certain equipment they use [1].

2-3-2- Integrity

Integrity defines the reliability of the data assets. It is usually designed in such a way to avoid misuse of an organization's data resources or unofficial alterations to the system. Usually, integrity incorporates both data (the content of the information) and origin (the data source) integrity. The latter is commonly known as authentication. The individual user rating and confidence on the information is the underlying measure for the accuracy and credibility of the source of the information. "This dichotomy illustrates the principle that the aspect of integrity known as credibility is central to the proper functioning of a system" [1].

Integrity mechanisms fall into two classes:

2-3-2-1- Prevention Mechanisms

This method makes sure unauthorized efforts to alter data resources are prevented so as to retain the integrity of the information. The process also blocks any activity that can lead to alterations of the data resources in an unauthorized manner. However, it's important to understand the difference between the two forms of attempts. The first attempt is applicable when an individual who has no official permission attempts to change information, whereas the second one happens to deny permitted user from executing certain alteration beyond his jurisdiction [1].

2-3-2-2- Detection Mechanisms

The detection mechanisms basically report when the information integrity cannot be trusted any more. Unlike the prevention mechanisms, the detection mechanisms do not attempt to block violation of integrity. However, it examines sequence of the system events in order to identify the problem. At the same time, it may check the data information to detect whether required or expected restrictions are still in place. These mechanisms could give detail

of the possible cause of the integrity violation. In some cases declares the file for being corrupted [1].

2-3-3- Availability

Availability of required resource or information is its ability to be used. This is a significant requirement of reliability and system design, as inaccessible system is no different to lack of the system in any ways. The organization may decide to make the data unavailable or consciously prevent the information access which is an aspect of availability. This phenomenon is vital to maintaining the computer security. The designed system normally assumes a statistic model in order probe expected pattern of use. This is vital to availability, as it guarantees availability of the data when specifications of the statistical model are met. However, an individual could be capable to influence the access, hence rendering invalid assumptions by the model. This therefore, causes failure of the system, as the method allowing availability of the data resources are operating outside its prescribed context [1].

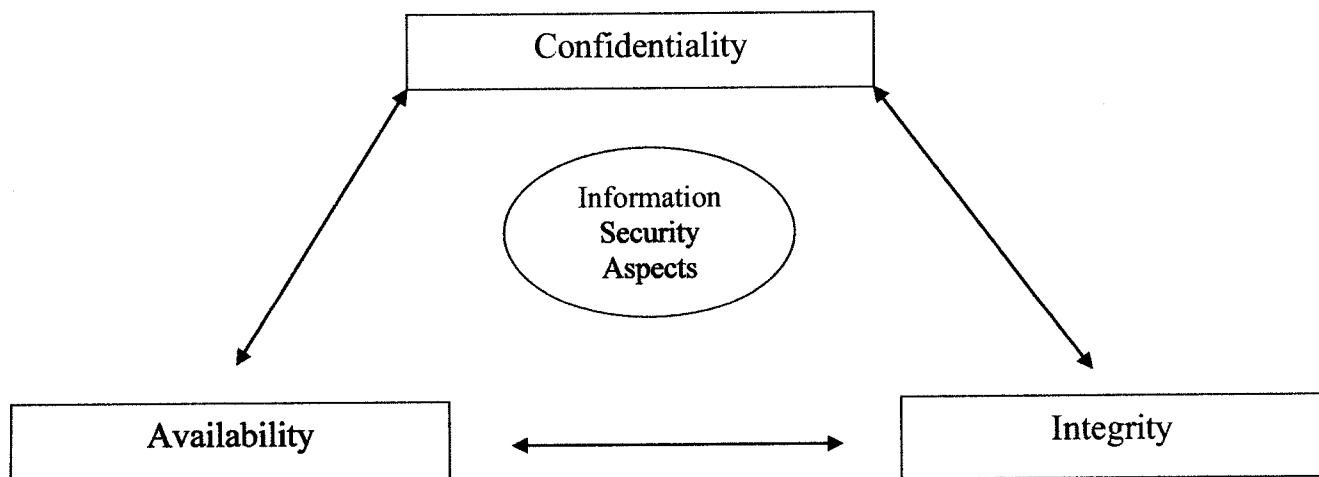


Figure 1: Information Security Aspects (adopted from: Russell and Gangemi, 1991)[12].

2-4- Security Policy

2-4-1- Definition

A security policy defines specified rules and regulation set to ensure communication is maintained between the staff managing the system and individual user. Security policy is expected to interpret and verify primary security. Hence security policy acts as link between the management's objective and the security requirement of the users [6].

2-4-2- Security Policy Models

2-4-2-1- Overview

A company's security policy models summarize statements of how organizational assets and properties are to be protracted, typically written in one page or less. Organizations typically state their protection goals of various systems as agreed by the senior executive management and hence functions as the basis of guidance of do and don'ts with regards to protection and maintenance of systems, assets and other sensitive organizational properties that are susceptible to compromise. There are different security models adopted by different organizations, each has merits and demerits and sometimes related to protection of assets on a specific industry. Below is a summary of some for the leading and popular models in the security policy [4].

2-4-2-2- Bell-LaPadula Policy Model

This model was adopted as a result of a research work conducted by David Elliott Bell and Leonard J. La Padula in the 1970s. This security model is an access control that uses formal and mathematical description approach to explicitly denote how a security should be maintained, by using some sort of comparison to determine an access right based on the authorization level of the requested and the level of access authorized on the object. There was a widespread realization that protection of commercial operating systems was

poor, bugs and system vulnerabilities kept being discovered on commercial systems as soon as one is fixed. As a result, US government introduces a reference monitor concept and this was adopted as a result of a study conducted by James Anderson who concluded that protection of secure systems and properties should be enforced by using a simplified verifiable mechanism that rarely change. Reference monitor is part of Trusted Computing Base (TCB). TCB is a set interrelated hardware, software human components that together ensures enforcement of a security policy failure of which causes a breach of security policy [6].

2-4-2-3- Clark-Wilson Model

This security policy model is widely used in commercial environments to primarily ensure integrity of organizational data and prevent corruption of data due to error or malicious intent. It was proposed by David D. Clark and David R. Wilson in 1987. The policy clearly defines how organizational data should be kept securely with data integrity intact. It distils centuries old double-entry bookkeeping method which tends to mitigate insider fraud and ensure integrity of bank's accounting system. By posting each transaction to two different books namely Credit and Debit (assets and liabilities respectively), the accounting book should balance anytime a reconciliation is done. This ensures information integrity. Enforcing this policy entails principle of splitting job responsibilities to more than one staff and hence any potential fraud may requires collusion of two or more staff [6].

2-4-2-4- Chinese Wall Model

Is a popular security model proposed by Brewer and Nash in 1989. This policy tends to prevent flow of information in situation where conflict of interest is prevalent. This policy is widely used by professional consultants and investment bankers to build a virtual wall that separates the control the flow of information in order to avoid a conflict of interest or insider trading in a financial setup. For example, a conflict of interest arises here of an individual consults for two companies of same industry, particularly in a financial sector, as soon as a consultant interacted with two companies of same time business line. A closed 'Chinese Wall' is erected once an individual consulted for companies of same industry type and hence avoids conflict of interest on the job [6].

2-5- Security Policy language

2-5-1- Overview

This is the language that represents the security policy. The security policy languages are categorized into two groups; High-level policy language and Low-policy language.

2-5-2- High-Level Policy Languages

These refer to the restrictions imposed on the system's entities and commands which provide defined terms of the security policy. The accuracy of such expression demands formulation mathematical model of the security where ordinary English could not be applied [1].

2-5-3- Low-Level Policy Languages

A low-level policy language refers to a collection of inputs, argument to command those inputs and to verify the limitations on the system [1].

2-6-Policy Management

Customary reviews have to be employed in order to make sure the security policy is up to date. This may be executed in such a way that the organization's operation systems are directly translated into the security policy.

It was suggested that a special unit such as data security unit be assigned to administer the security policy. The unit would be in charge of performing expected reviews and keeping the security policy up to dated [6].

Killmeyer (2006) outlined the duty of security management. He stated the security management should provide direction and put procedures in place to control and minimize the risk of losing data and information. He also stressed that the security management requires further security measures in order to identify the origin of all message coming through and their authenticity. When the messages are strictly authenticated, that will help to reinforce security, and stop any attempt to forget identities [10].

2-7-Repudiation

Killmeyer also discusses the concept of repudiation. Non repudiation means receiving a proof of delivery and certification of the source. This is a method which guarantees that the sender cannot later deny that he/she has sent the message and also the recipient cannot refuse that he/she has received the message. This is an alternative way to manage and apply security by monitoring the messages going through the networking system[10].

2-8-Development of security policy

Security policy is significant in ensuring efficient and understandable security systems are created. Although the essential nature of developing security policy is frequently ignored, it serves as a vital element of the computer security design. The main purpose of the security policy is to decode the management's requirements for the security into a simple well defined and directed to a precise targets or objectives. Using a top down method is essential to derive an efficient and well developed overall security architecture. In the other hand, lack of security policy to relay the management's expectations will result in such decisions being made directly by individual users. Therefore significant operations such as installation and maintaining the computer system will be left in the hands of the individual users to be made. These consequently would lead to the implementation of an ineffective computer security systems or architecture [6].

2-9-Security policy violation

Violation of the security policy is a very serious threat to an organization's data assets. The violation may not necessarily take place in order to pose a risk at an organization's security resources. However it's essential to protect against those activities that may lead to the security violation. The actions that cause violation are commonly known as attacks which are usually carried out by attackers [2].

2-9-1-Interception

Interception is an unlawful disclosure of information which is a serious breach and a risk to confidentiality. This is mostly carried out through passive wiretapping which happens when listening to certain entity, reading or browsing files or systems information. The threat is also referred to as snooping or eavesdropping [2].

2-9-2-Modification/Alteration:

This is an unlawful alteration of the data security assets which is a threat to the data reliability. As opposed to interception, modification is an active form of violation. This might be achieved by active wiretapping, which is an alteration that influences the nature of data communication between networks. A typical example is the man in middle attack where an intruder monitors and reads information from sender and then sends a modified edition to the recipient. In most cases, the presence of a third-party is not felt by either the sender or the reader [2].

2-9-3- Masquerading/Spoofing

Another threat to the data security reliability includes the violation such as masquerading or spoofing, which involves masquerade of an individual by another person. In this case, the violators disguise the true identity of the service from the individuals at the receiving end so as to entice him accept a particular entity [2].

2-9-4- Interruption

Threats to the data availability include a form of infringements such as “interruption”. This mainly takes place in a form of an unlawful denial of users’ access into specific data resource. A common example is revealed when an intruder influences the server

performing its service; a breach known as the denial-of-service attack. The attackers accomplish this through various mechanisms such as denial at the source which stops the server from receiving the information supply. These data resources are required for effective function of the server at its local environment thus hindering information transmission via the server across networks. The denial may also occur on the route of the intermediate pathway thereby getting rid of the information from the server or the client or influencing both of them [2].

2-9-5-Fabrications

Fabrications are another form of violation which compromises the security integrity. These involve assembly of forged resources or unlawful addition of information. A common example is the insertion of false information in a network [2]. However, the staffs are required to develop complete understanding and knowledge of the costs of the policy violation, so as to be able to appreciate the significance of the security policy.

Security policy violation is a serious threat to an organization's corporate systems and should be dealt appropriately. Staff or individuals who violated the policy must be notified about their action and cautioned to face penalty. These may be done by placing them on a "trial period" by allowing them access only limited resources until they demonstrate competency to utilize the organization information in a safe way and comply with the security policy when using the corporate systems. In a more serious case of violation, the violator should be warned of being sacked or getting prosecuted. Although the latter punishment may seem to be too much, reasonable step has to be taken in any case of violation following the terms or guidelines as outlined in the AUP and the policy. However, the main concentration should be on maintaining the security rather than just the punishment for the violation. Without this, it would be difficult to penetrate which might be as a result of human mistake or misinterpretation of the security policy [5].

A web-based policy editor for SANTA	العنوان:
Al Mutairi, Abd Algder	المؤلف الرئيسي:
Siewe, Francois(Supervisor)	مؤلفين آخرين:
2009	التاريخ الميلادي:
Leicester	موقع:
1 - 105	الصفحات:
687206	رقم MD:
رسائل جامعية	نوع المحتوى:
English	اللغة:
رسالة ماجستير	الدرجة العلمية:
De Montfort University	الجامعة:
Faculty of Computing Sciences and Engineering	الكلية:
بريطانيا	الدولة:
Dissertations	قواعد المعلومات:
الأمن المعلوماتي، هندسة البرمجيات، الإنترنت، لغة البرامج	مواضيع:
https://search.mandumah.com/Record/687206	رابط:

Chapter 3: SANTA Language

3-1- Overview

The chapter introduces a security policy language commonly known as the SANTA. We would also look into the policy in SANTA by unfolding the authorization, obligation and delegation.

3-2- Definition

SANTA (Security Analysis Toolkit for Agent) is language for specifying security policies such as authorisation and obligation policies. SANTA is developed in the Software Technology Research Laboratory (STRL) at DMU and is used to specify and analyse security policies using a run-time verification tool called Tempura.

3-3-Policies in SANTA

A security policy in SANTA is based on the notion of subject, object and action. A subject is assumed to be an active entity that can perform an action on an object or another subject. Examples of subject are users, processes or agents running on behalf of users. An object is a passive entity that stores some data, e.g. a file, a storage device or a profile. An action is an operation that can be performed on objects or subjects, e.g. read a file, append to a file, withdraw money from a bank account and so on.

3-4- Syntax of SANTA Policy

3-4-1- Simple Condition

The syntax of the simple condition is that

<code>< cond > ::= < exp1 > < rop > < exp2 ></code>	relational conditions
<code>< rop > ::= = > < ≤ ≥</code>	relational operators
<code>< exp > ::= < var ></code>	variable symbols
<code> < const ></code>	constant symbols such as true, false, 0, 1, 2
<code> - < exp ></code>	unary minus
<code> (< exp >)</code>	parentheses
<code> < exp1 > < aop > < exp2 ></code>	binary arithmetic operations
<code>< aop > ::= + - * /</code>	arithmetic operators

Example of simple conditions:

- $x = 5$
- $x + y > -1$
- $2 * (x + 3) \leq 0$

3-4-2- Compound Condition

The syntax of the Compound condition is that

$\langle \text{Ccond} \rangle ::= \langle \text{cond} \rangle$	simple condition
$ \langle \text{Ccond1} \rangle \wedge \langle \text{Ccond2} \rangle$	conjunction
$ \langle \text{Ccond1} \rangle \vee \langle \text{Ccond2} \rangle$	disjunction
$ \neg \langle \text{Ccond} \rangle$	negation
$ (\langle \text{Ccond} \rangle)$	parentheses

Example of compound conditions:

- $x = 5$
- $(x + y > -1) \wedge \neg (x \leq 0)$

3-4-3- Policy Rule

3-4-3-1- Authorization Rule

This rule specifies access control measures. Authorization involves three types of rules namely positive authorization, negative authorization and decision rules[3].

Positive Authorization

These are declarations that express specific conditions under which specific summon for access could be tolerated. However, as far as ultimate decision of the policy for the right of system entry is concerned, only the signal is the taken into consideration [3].

A positive authorization rule has the form

$$f \longrightarrow \text{autho}^+(X, Y, Z)$$

The letter f represents ITL formula in this syntax. X,Y,Z represent subject, object and action respectively. The rule expresses that the subject X is clearly granted to carry out

the action Z on the object Y each time the specifications of f is fulfilled. The formula f is regarded as the premises or the body of the rule whereas the conclusion of the rule refers to the expression $\text{autho}^+(X, Y, Z)$ [2].

Negative Authorization

On the other hand, negative authorization rules describe condition under which the right of entry into a system should be denied. Although negative authorization operates in opposite direction to the positive authorization but the mechanism of implementing their roles are similar. This is due to the fact that under negative authorization as well, only the indication is place into consideration as far as the final access decision of the policy is concerned [3]. A negative authorization rule has the form

$$f \longmapsto \text{autho}^-(X, Y, Z)$$

The letter f represents ITL formula in this syntax. X,Y,Z represent subject, object and action respectively. The rule expresses that the subject X is clearly rejected from carrying out the action Z on the object Y each time the specifications of f is satisfied

Conflict resolution rules

These are rules that indicate the ultimate decision of access regulation of the policy. It is necessity to incorporate decision rule into any policy, without which access would permanently be denied.

A conflict resolution rule has the form

$$f \longmapsto \text{autho}(X, Y, Z)$$

This rule was derived in order to express specifications that define the right of access for both positive and negative authorizations. The variance within the authorization rules are fixed by the application of the conflict resolution rules [2]

3-4-4- Simple Policy

This refers to a set of policy rules. The syntax below is usually applied to define the behaviour of the simple policy. The letters p and q ranges over simple policy whereas r refers to a policy rule.

$$p ::= \emptyset \mid r \mid p \cup q \mid p \cap q \mid p \setminus q \mid \neg p \mid \pi^+(p) \mid \pi^-(p)$$

"The empty policy \emptyset contains no rules. The set operators "U", " \cap " and " \setminus " have their usual meaning. In particular, the operator "U" can be used to add rules to a policy while the operator " \setminus " can be used remove rules from a policy. The unary operator " \neg " changes positive authorisation rules into negative ones and negative authorisation rules into positive ones and leaves conflict resolution rules unchanged if there is any in the policy argument. The operators π^+ and π^- return respectively the set of positive authorisation rules and the set of negative authorisation rules contained in the policy argument" [2].

3-4-5- Compound Policy

Compound policy is a composition of simple policies.

The syntax is the following:

$\langle policy \rangle ::= \langle simple\ policy \rangle$	simple policy
$\mid \{ \langle Ccond \rangle \} \langle policy \rangle$	unless
$\mid [\langle Ccond \rangle] \langle policy \rangle$	as long as
$\mid \langle const \rangle : \langle policy \rangle$	duration
$\mid \langle policy1 \rangle ; \langle policy2 \rangle$	chop

3-4-6- Delegation

The process by which one subject passes on a specific access privileges to another different subject is known as delegation. This is a very effective model that permits one off shift of right of a system access to another subject. The policies on delegation are there to allow responsible use of such concept, and at the same time, lessening the authority of the subject to undertake the practice [3].

A delegation has the form

$$\text{deleg} \stackrel{\wedge}{=} \{ \text{deleg}(s_1, s_2, o, a) \mid s_1, s_2 \in S, o \in O, a \in A \}$$

In analogy to the authorisation model, delegation is modelled using Boolean variable. The Boolean states variable used in such a way that the access right to execute action a on object o id delegated from subject s1 to s2 [2].

3-4- Obligation

This refers to the specific set of terms and conditions under which a subject has an obligation to carry out the action on an object. The specification or conditions of the obligation should be satisfied by the obliged subject. This process minimizes the implementation of an obligation on the protection discharge mechanisms [3].

3-5- Types of policies in SANTA

SANTA policies are categorized into two: Environmental policy and behavioral policy.

- Environmental policies:

The main purpose of the environmental policies is to maintain integrity of an object as well as control access to the data resources of a system [3].

- Behavioral policies

These are policies that control the actions or behavior of a subject. The behavioral policies are designed in such a way that to outline limitation the user rather favors to be implemented [3].

3- 6- Policy specification

This is a mechanism by which non-official security enforcement specifications in the policy language are demonstrated. The rules in the policy act as the foundation for SANTA policy requirements development. However, devising set of rules that essentially govern the non formal requirements is among most essential needs during the specification development [3].

3-7- policy Composition

In SANTA policies, a sophisticated set of various operators can be implemented to derive specified policies in small entities which is more advantageous compared to the other policy languages [3].

3-8-Summary

In this chapter we explained the SANTA language as policy language. SANTA is a language for specifying security policies such as authorization and obligation policies. These policies are based on the notion of Subject, Object and Action. A subject is assuming to be an active entity that can perform an action on an object or another subject. An object is a passive entity that stores some data, e.g. a file, a storage device or a profile. An action is an operation that can be performed on objects of subjects, e.g. read a file, append to a file, withdraw money from a bank account and so on.

A web-based policy editor for SANTA	العنوان:
Al Mutairi, Abd Algder	المؤلف الرئيسي:
Siewe, Francois(Supervisor)	مؤلفين آخرين:
2009	التاريخ الميلادي:
Leicester	موقع:
1 - 105	الصفحات:
687206	رقم MD:
رسائل جامعية	نوع المحتوى:
English	اللغة:
رسالة ماجستير	الدرجة العلمية:
De Montfort University	الجامعة:
Faculty of Computing Sciences and Engineering	الكلية:
بريطانيا	الدولة:
Dissertations	قواعد المعلومات:
الأمن المعلوماتي، هندسة البرمجيات، الإنترنت، لغة البرامج	مواضيع:
https://search.mandumah.com/Record/687206	رابط:

Chapter 4: Modelling of the System By (FSM)

4-1- Introduction

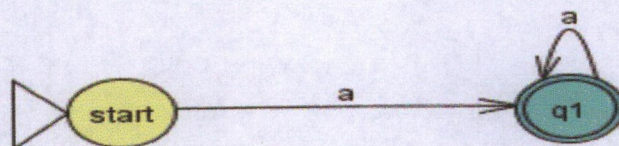
A Finite State Machine (FSM) method would be applied here to symbolize the model of system. This will enable to demonstrate the mechanism of the system operation in the early stages.

4-2-Definition of Finite State Machine

Hierons described finite state machine (FSM) “ is model that defines the required behaviour of an implementation, it is important to verify the implementation against the FSM” [7].

A finite state machine (FSM) is formal method for modeling system to help the programmers to understand the system. In this stage we will use the JFLAB tool to create all states and transition between each state [8].

This is an example for basic model of finite state machine which shows two states and between them a transition denoted with letter “a” in the diagram below.



Figuer2: Basic model of (FSM)

4-3-JFLAP

JFLAP is a package of graphical tools which can be used as an aid in learning the basic concepts of Formal Languages and Automata Theory [7].

4-4- Design Model of State

Under this section, a formal state model would be designed in order to facilitate comprehension of the high level system analysis and development. It will also provide easy use of the technical skills and application of transferable skills in communication.

4-4-1- Simple condition model

Simple condition syntax

$\langle \text{cond} \rangle ::= \langle \text{exp1} \rangle \langle \text{rop} \rangle \langle \text{exp2} \rangle$ relational conditions

$\langle \text{rop} \rangle ::= = | < | > | \leq | \geq$ relational operators

$\langle \text{exp} \rangle ::= \langle \text{var} \rangle$ variable symbols

| $\langle \text{const} \rangle$ constant symbols such as true, false, 0, 1, 2

| $- \langle \text{exp} \rangle$ unary minus

| $(\langle \text{exp} \rangle)$ parentheses

| $\langle \text{exp1} \rangle \langle \text{aop} \rangle \langle \text{exp2} \rangle$ binary arithmetic operations

$\langle \text{aop} \rangle ::= + | - | * | \backslash$ arithmetic operators

Example of simple conditions:

$x = 5$

$x + y > -1$

$2 * (x + 3) \leq 0$

The first model we designed is simple condition. It has eight states and each state has different transition.

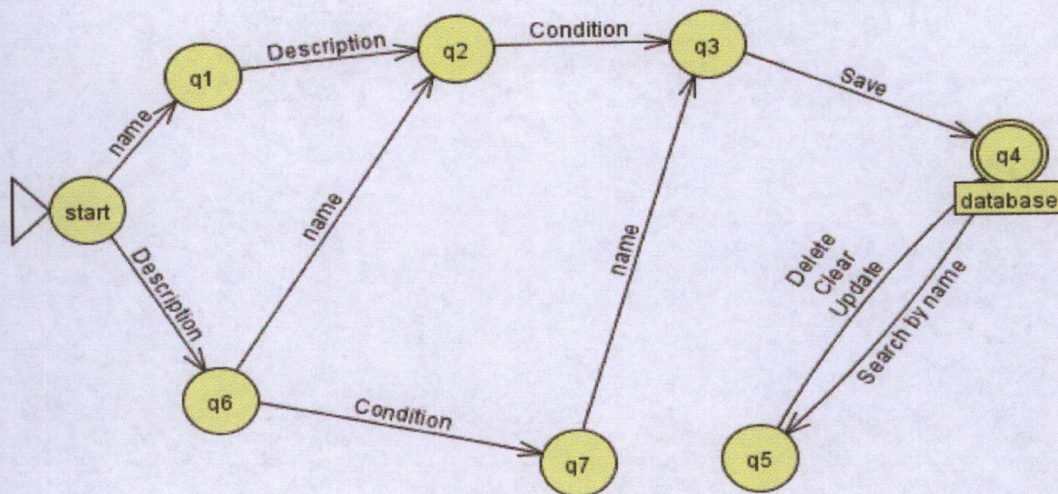


Figure3 Simple Condition Model (FSM)

It can be seen from this model that the user can create simple condition and save it in database. So the user can not save the simple condition without name it as a result the system will reject. In addition, he can do other functionality such as delete, update and search.

4-4-2- Compound Condition Model.

Compound condition is consists of one simple condition or more. It has a different model from simple condition.

Compound Condition syntax:

< Ccond > ::= < cond >	simple condition
< Ccond1 > ^ < Ccond2 >	conjunction
< Ccond1 > v < Ccond2 >	disjunction
¬ < Ccond >	negation
(< Ccond >)	parentheses

Example of compound conditions:

X=5
 $(x + y > -1) \wedge \neg(x \leq 0)$

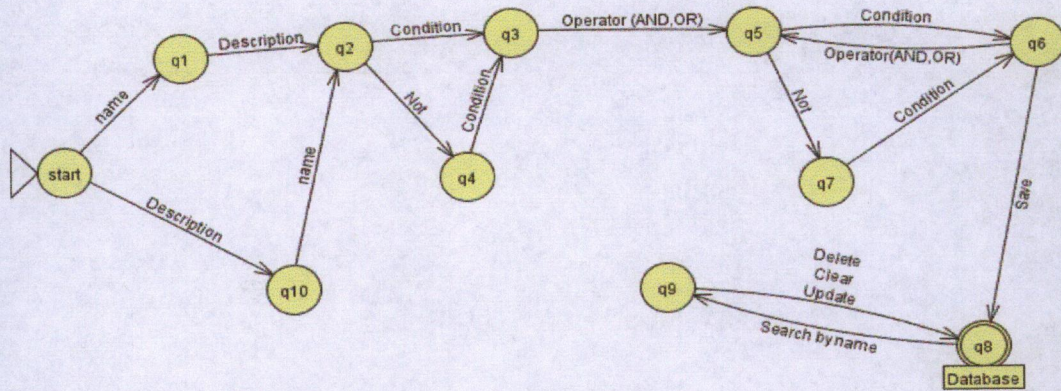


Figure 4: Compound Condition Model (FSM)

This model shows that the user can create compound condition and save it in database. The system restricts the user which prevents him to save invalid format for example if the user put at the end of the condition operator.

4-4-3- Policy Rule Model

Policy rule consists of two parts premise and consequences (type of authorisation, subject, object and action). This model will represent how policy rule works.

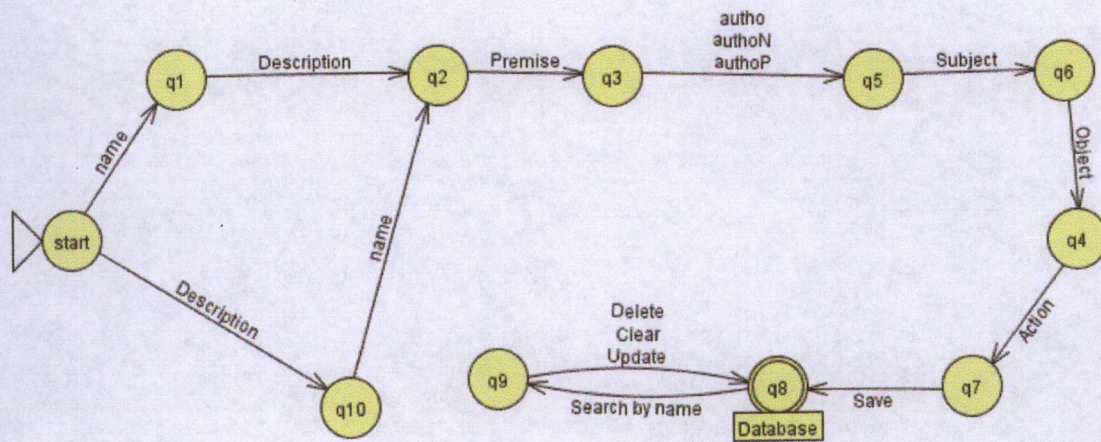


Figure 5: Policy Rule Model (FSM)

From this model it can be seen that policy rule has 11 states and final state is q8 which is database, that user can save his policy rule. The user can also choose any type of access (authoP,authoN,autho).

4-4-4- Simple Policy Model

Simple policy is set of rules and in this model we will show how the user can create the policy.

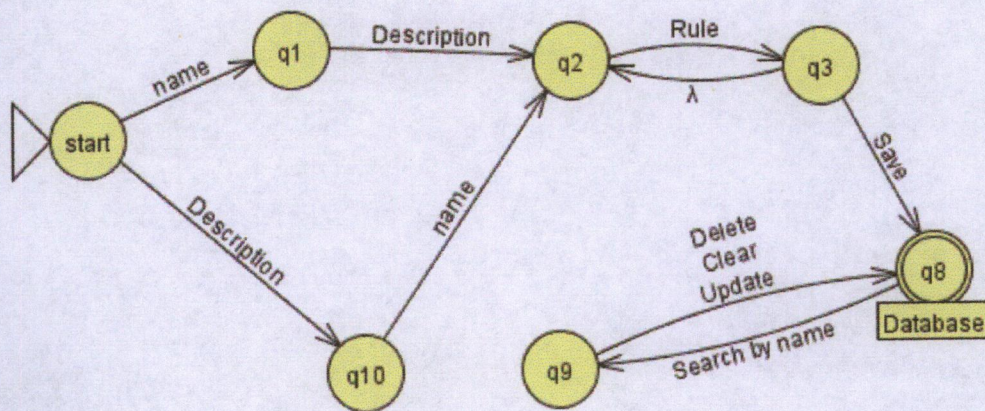


Figure 6: Simple Policy Model (FSM)

This model shows that the user can create his simple policy by choosing one rule or more than one. This is obvious from the iterative transition from q3 to q2 to let the user to choose another one . It also strict the user to write the name otherwise he can not save his policy.

4-4-5- Compound Policy 1

Compound Policy is composition of simple policies. So in this report we will divide that in two steps, the first one is compound policy1 and the second is compound policy2. In compound Policy1 interface user can create one of these elements (simple policy, as long as, unless, duration). Then he can save it as compound policy1 which will use it later in compound policy2.

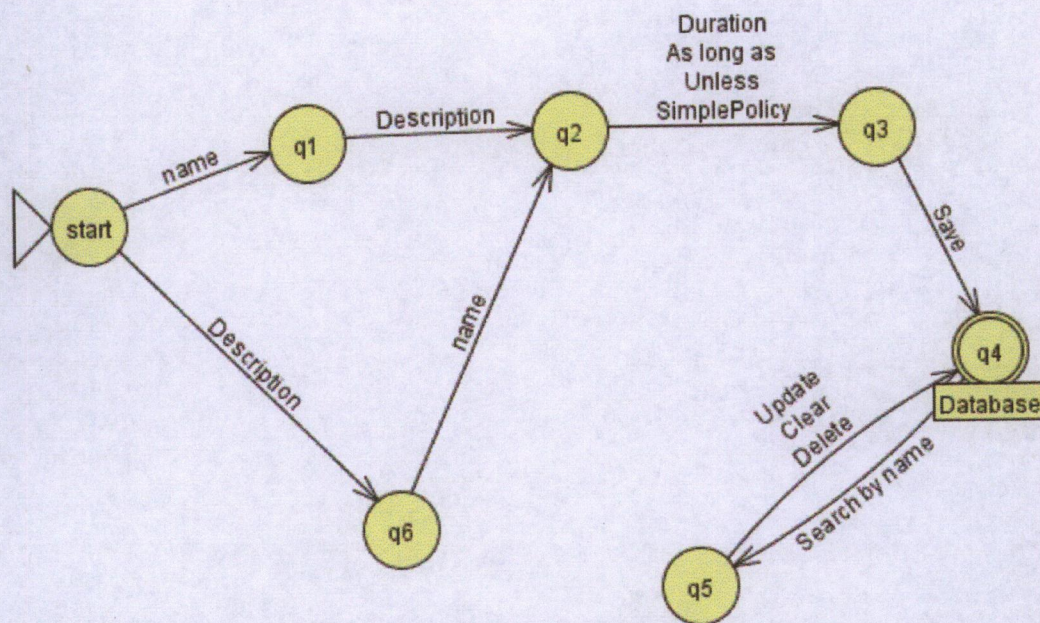


Figure 7: Compound Policy1 Model (FSM)

The model shows that there are seven states and the first state is start, final state is q4. The system strict user by letting him to create only one element to move from q2 to q3.

4-4-6- Compound Policy 2 Model

Compound policy2 is composition of compound policy 1 which is created early.

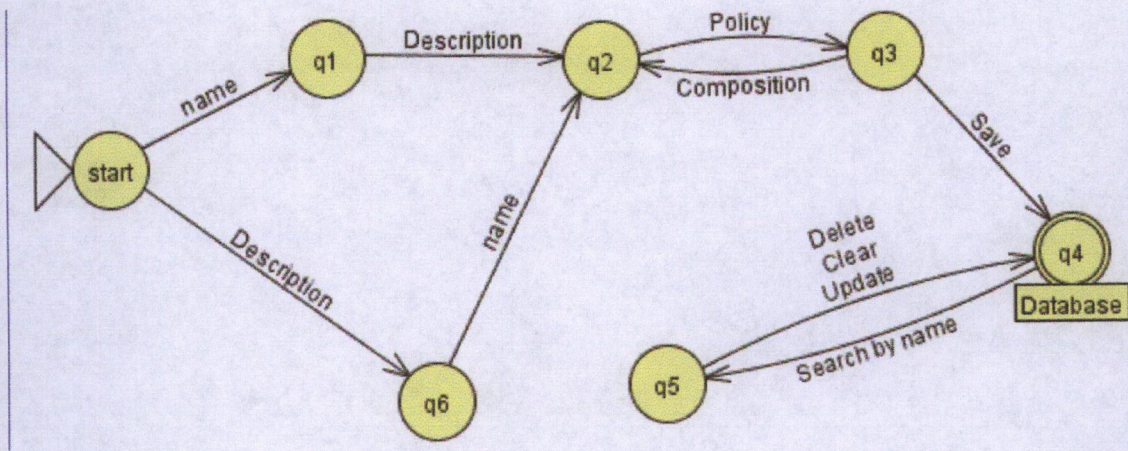


Figure 8: Compound Policy2 Model (FSM)

From this model it can be seen that the user can make composition for his policies. In This model also he can create one policy or more but it must be made composition If it is more than one.

4-4-7- Subject and Object Model

There is no different between Subject and Object model all the structure. So we design one model to show the structure for both of them.

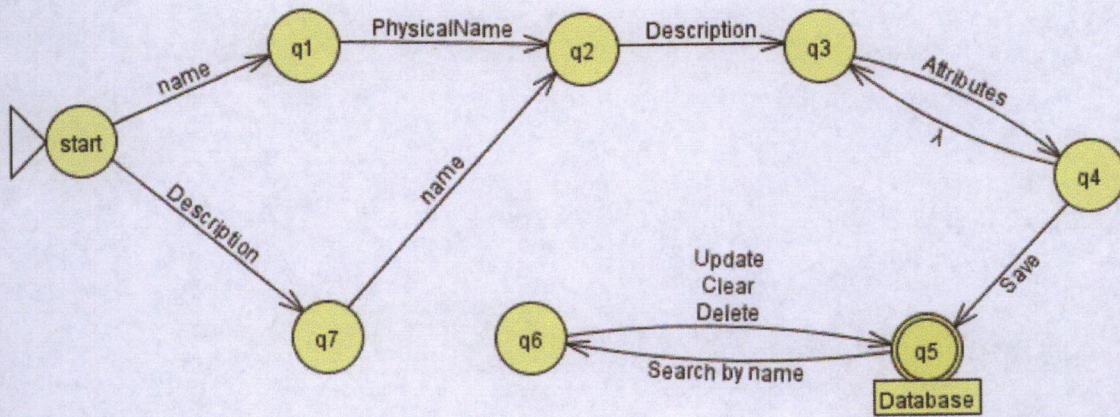


Figure 9: Subject and Object Model (FSM)

This model shows that the user can build the subject and object with their attributes. The system can create more than one attribute for each entity.

4-4-8- Action Model

This Model is different model from the previous model because it has different structure.

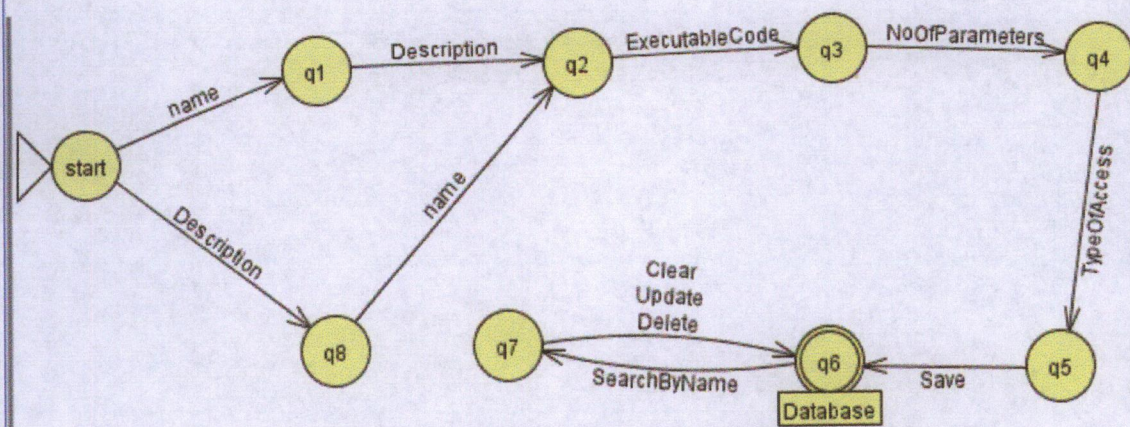


Figure 10: Action Model (FSM)

4-5- Testing the state machine

4-5.1. Overview

In this division we will test all model to make sure if models were created properly. So we will use all possible tests.

4-5.2. Type of testing using JFLAP

The JFLAP has different type of testing which is helpful such as test by step state, fast run and multiple run. We will apply each type for different model.

4-5-3- Result of the test

Will be testing the simple condition by using multiple Run:

4-5-3-1-Multiple Run

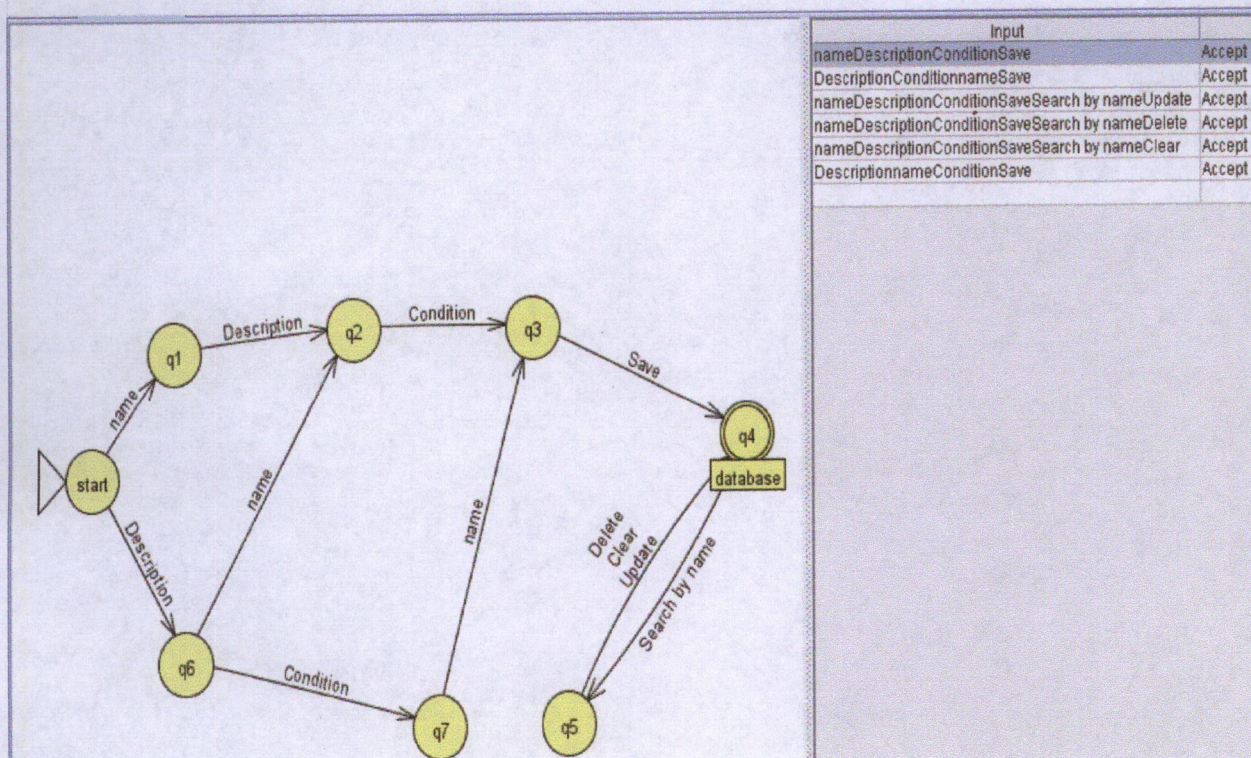


Figure 11: Simple condition model test

4-5-3-2- Step by state

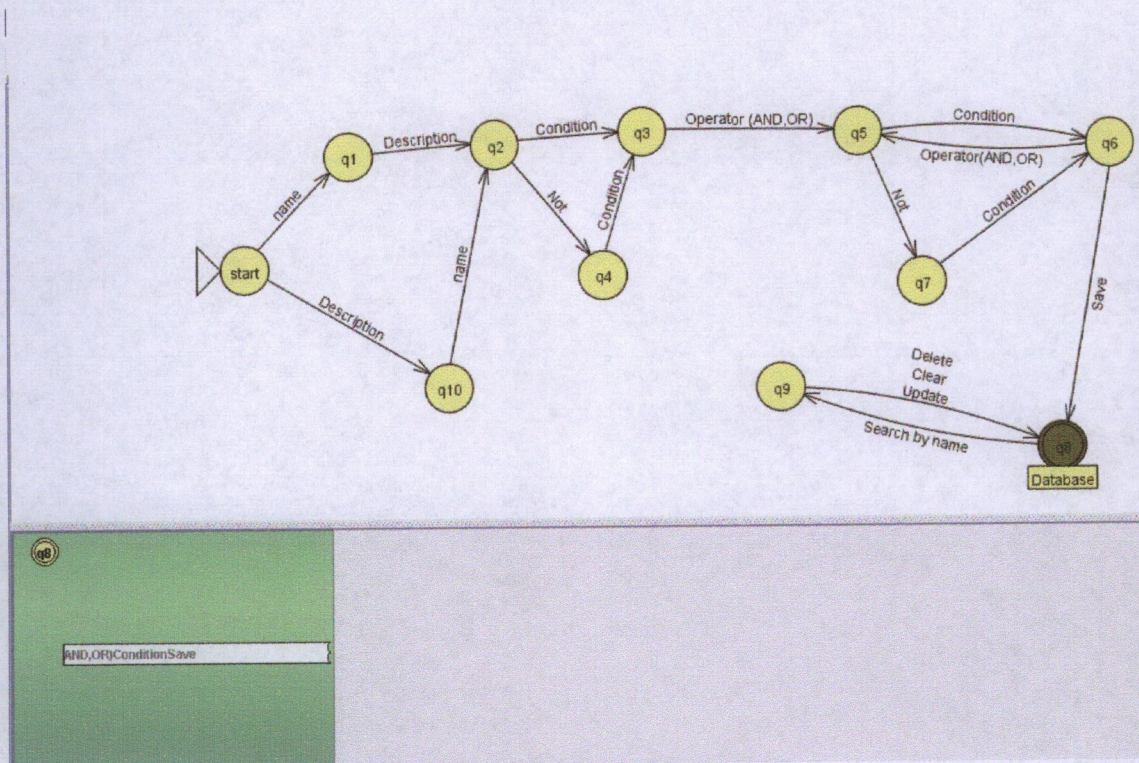


Figure 12: Compound Condition Model test

4-5-3-3- fast test

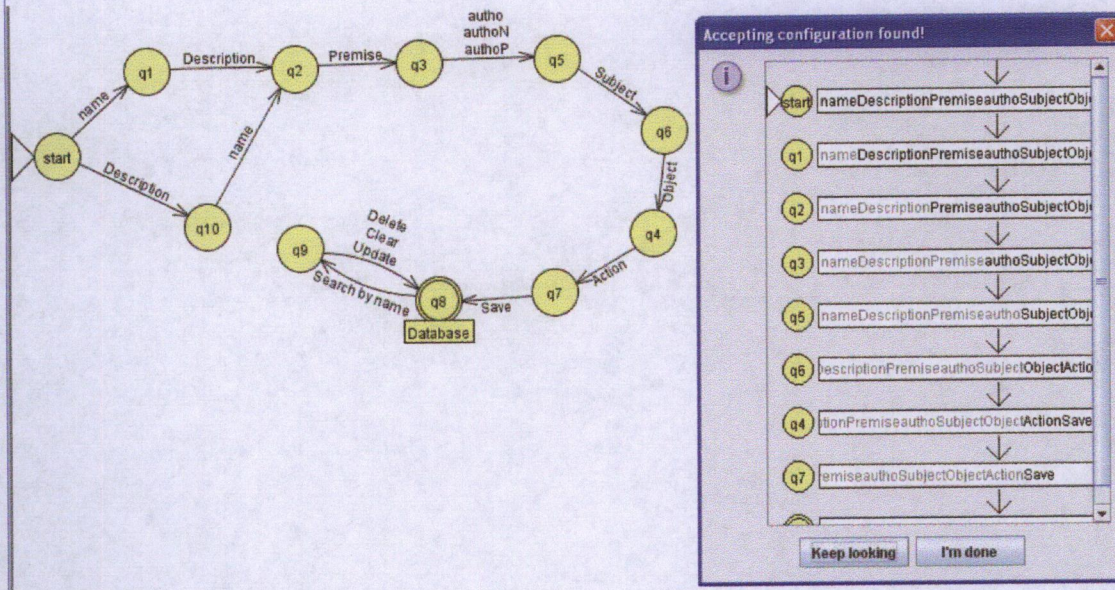


Figure 13: Action Model (FSM)

4-6-Summary:

In this chapter we tried to model the system by using the FMS and JFLAB because they are very significant method to model the system. So the programmers focus on the system modeling to understand how it works. In addition, they ensure if the functionality of the system work as specified or not. The benefit of that is to help the programmers to discover problems and fit them in early stage.

A web-based policy editor for SANTA	العنوان:
Al Mutairi, Abd Algder	المؤلف الرئيسي:
Siewe, Francois(Supervisor)	مؤلفين آخرين:
2009	التاريخ الميلادي:
Leicester	موقع:
1 - 105	الصفحات:
687206	رقم MD:
رسائل جامعية	نوع المحتوى:
English	اللغة:
رسالة ماجستير	الدرجة العلمية:
De Montfort University	الجامعة:
Faculty of Computing Sciences and Engineering	الكلية:
بريطانيا	الدولة:
Dissertations	قواعد المعلومات:
الأمن المعلوماتي، هندسة البرمجيات، الإنترنت، لغة البرامج	مواضيع:
https://search.mandumah.com/Record/687206	رابط:

Chapter 5: Requirement, Analysis and Design

5-1- Overview:

In this chapter, requirements are analyzed and system design is represented. We will use the UML (Unified Modeling Language) notation to analyze the system requirements and to design the system. This design will lead to the implementation of the working system. All the system stakeholders are identified and proper collaborations/interactions are presented [13].

5-2- Problem Domain Requirements:

“The first stage addresses the highest level (or widest scope) of the requirements analysis. It is in terms of this level that other levels of analysis, down to the reportable attributes which actually express measurements of systems, must be validated “[13].

“The basic tasks the system is required to address are functional requirements, which at the top level of description should relate as closely as possible to valid and measurable user requirements”[15]. For Policy Editor for SANTA, “this is relatively straightforward, since the main functional requirements can be expressed in terms of facilitating the end user to perform the main intended activities under certain condition”.[15]

5-3-System Requirements:

Policy Editor for SANTA must allow the user to perform the following tasks.

- Add Update, Delete and Search the Subjects, Objects and Actions.
- Add Update, Delete and Search the Simple Conditions and Compound Conditions.
- Add Update, Delete and Search the Simple Policies and Compound Policies.

5-4- UML Diagrams:

5-4-1-Use Case Model:

A use-case model represents the actors, the use cases and the relationship between them. The actors are who represent the interaction with the system and exchange information with the system. When these actors interact with the system the system perform some actions/activities based on the information exchanged by the actor and provides the desired results/functionality to the actor/user. Actor is always external to the system which on interact with the system using the provided interfaces [13].

5-4-1-1 System Level Use Case Diagram:

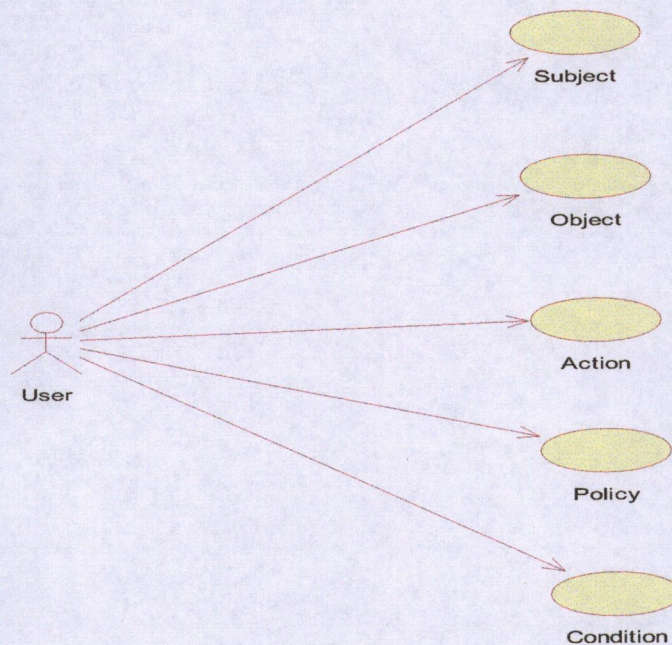


Figure 14: System level

System Level Use Case Description:

The main actor of the system is any non-specialist user who will interact with the system to manage or create the Subject, Objects, Actions, Policies and Conditions. System will perform some actions against the actor's request to fulfil the requirement.

5-4-1-2-Elaborated Use Cases:

Action:

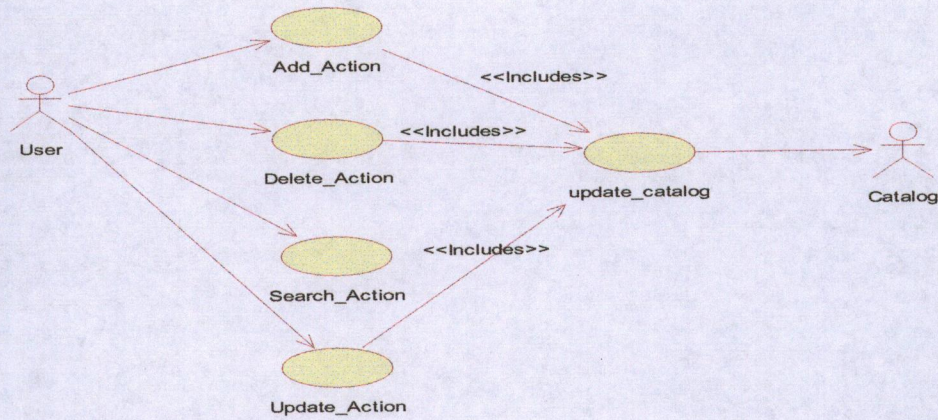


Figure 15: Action Use Case Diagram

Action Use Case Description:

Goal: User will be able to Add/Delete/Update/Search the action.

Actors: Non-Specialist User, System Catalogue.

Pre-Conditions: User gets the refreshed page to add new actions. To delete, update and search an action is already saved to the system catalogue.

Post-Condition: Action is Added/Updated/Deleted/Searched successfully.

Subject:

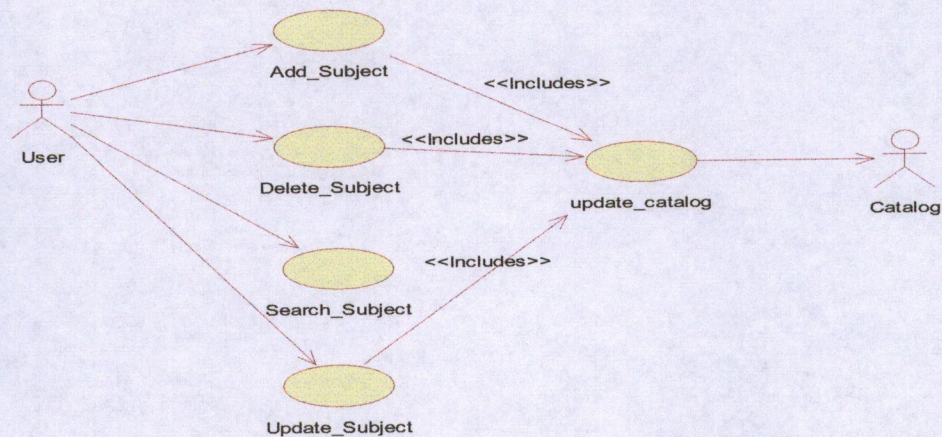


Figure 16: Subject Use Case Diagram

Subject Use Case Description:

Goal: User will be able to Add/Delete/Update/Search the subject.

Actors: Non-Specialist User, System Catalogue.

Pre-Conditions: User gets the refreshed page to add new actions. To delete, update and search a subject is already saved to the system catalogue.

Post-Condition: Subject is Added/Updated/Deleted/Searched successfully.

Object:

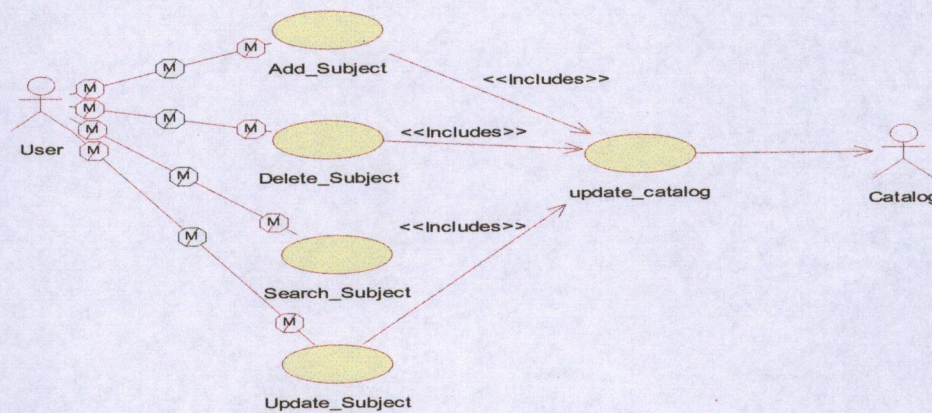


Figure 17: Object Use Case Diagram

Object Use Case Description:

Goal: User will be able to Add/Delete/Update/Search the object.

Actors: Non-Specialist User, System Catalogue.

Pre-Conditions: User gets the refreshed page to add new actions. To delete, update and search an object is already saved to the system catalogue.

Post-Condition: Object is Added/Updated/Deleted/Searched successfully.

Policy Rule:

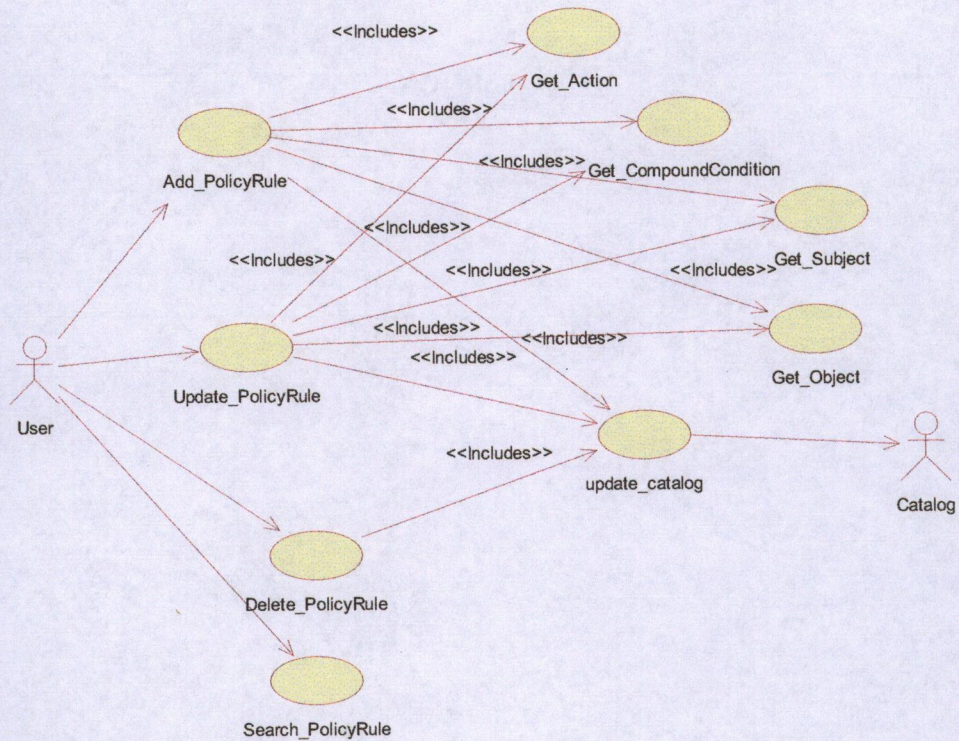


Figure 18: Policy Rule Use Case Diagram

Policy Rule Use Case Description:

Goal: User will be able to Add/Delete/Update/Search the Policy Rule.

Actors: Non-Specialist User, System Catalogue.

Pre-Conditions: User gets the refreshed page to add new actions. To delete, update and search a Policy Rule is already saved to the system catalogue.

Post-Condition: Policy Rule is Added/Updated/Deleted/ Searched successfully.

Simple Policy:

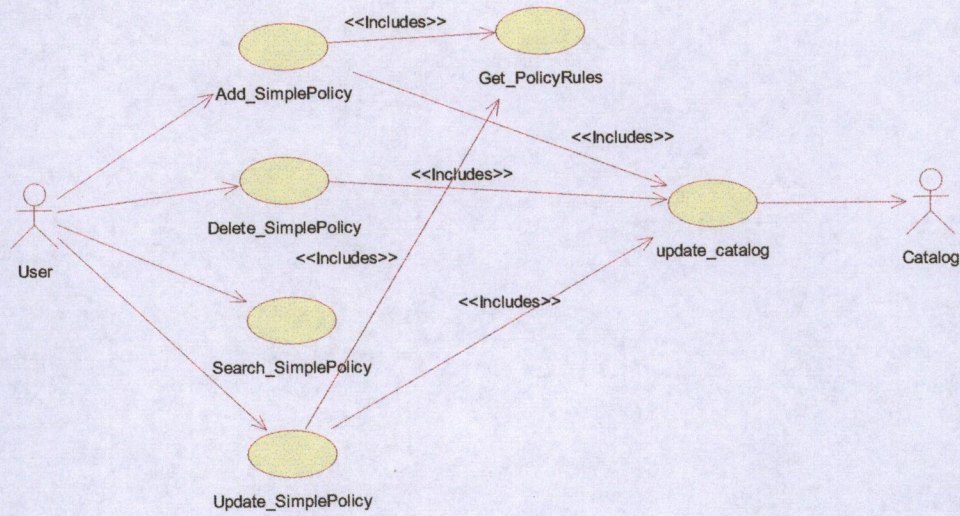


Figure 19: Simple Policy Use Case Diagram

Simple Policy Use Case Description:

Goal: User will be able to Add/Delete/Update/Search the Simple Policy.

Actors: Non-Specialist User, System Catalogue.

Pre-Conditions: User gets the refreshed page to add new actions. To delete, update and search a Simple Policy is already saved to the system catalogue.

Post-Condition: Simple Policy is Added/Updated/Deleted/ Searched successfully.

Compound Policy1:

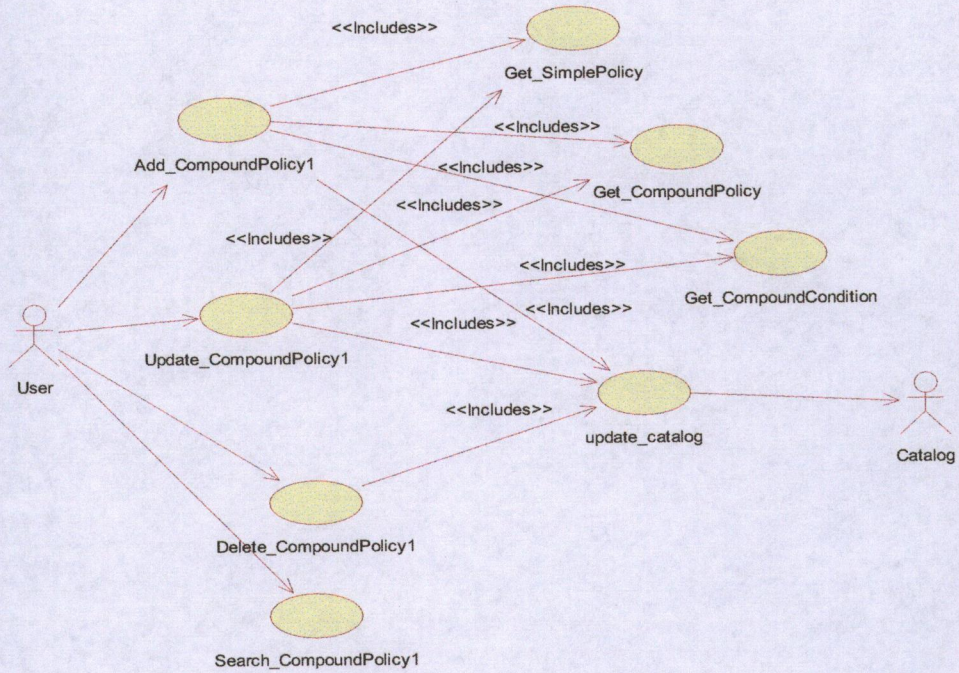


Figure 20: Compound Policy1 Use Case Diagram

Compound Policy1 Use Case Description:

Goal: User will be able to Add/Delete/Update/Search the Compound Policy1.

Actors: Non-Specialist User, System Catalogue.

Pre-Conditions: User gets the refreshed page to add new actions. To delete, update and search a Compound Policy1 is already saved to the system catalogue.

Post-Condition: Compound Policy is Added/Updated/Deleted/Searched successfully.

Compound Policy2:

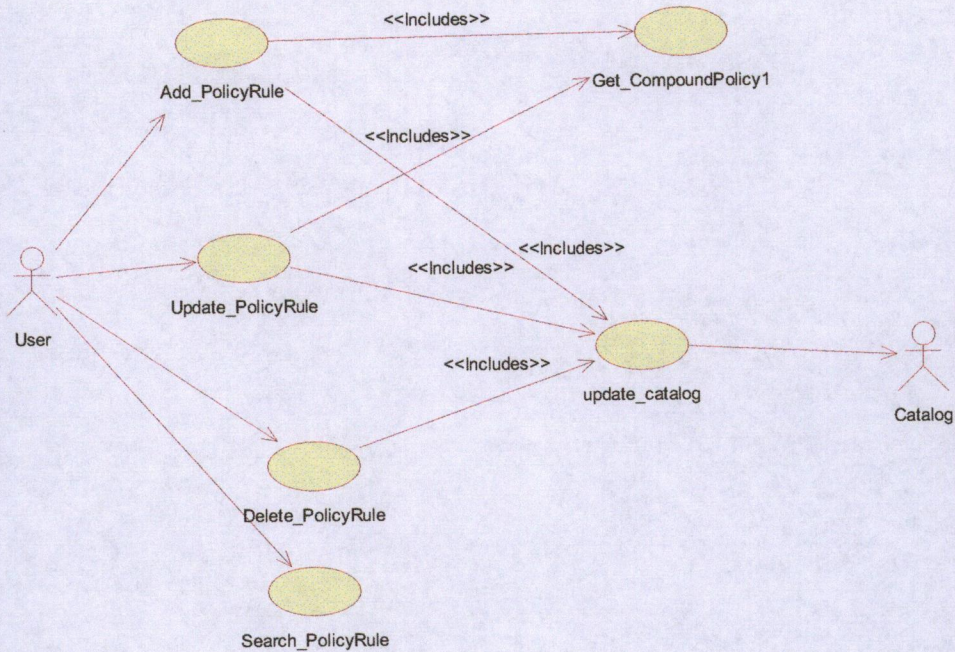


Figure 21: Compound Policy2 Use Case Diagram

Compound Policy2 Use Case Description:

Goal: User will be able to Add/Delete/Update/Search the Compound Policy2.

Actors: Non-Specialist User, System Catalogue.

Pre-Conditions: User gets the refreshed page to add new actions. To delete, update and search a Compound Policy2 is already saved to the system catalogue.

Post-Condition: Compound Policy is Added/Updated/Deleted/Searched successfully.

Simple Condition:

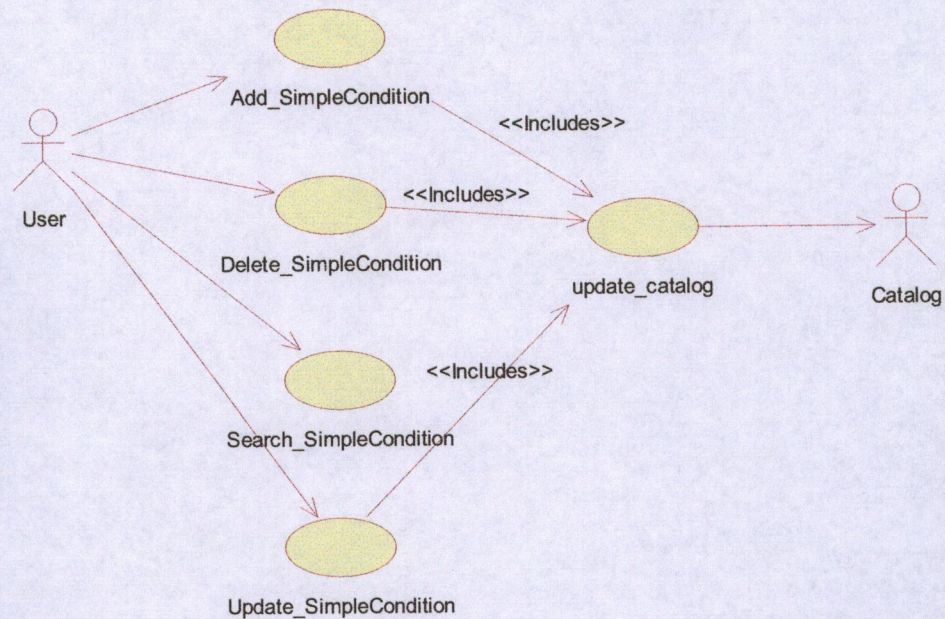


Figure 22: Simple Condition Use Case Diagram

Simple Condition Use Case Description:

Goal: User will be able to Add/Delete/Update/Search the Simple Condition.

Actors: Non-Specialist User, System Catalogue.

Pre-Conditions: User gets the refreshed page to add new actions. To delete, update and search a Simple Condition is already saved to the system catalogue.

Post-Condition: Simple Condition is Added/Updated/Deleted/Searching successfully.

Compound Condition:

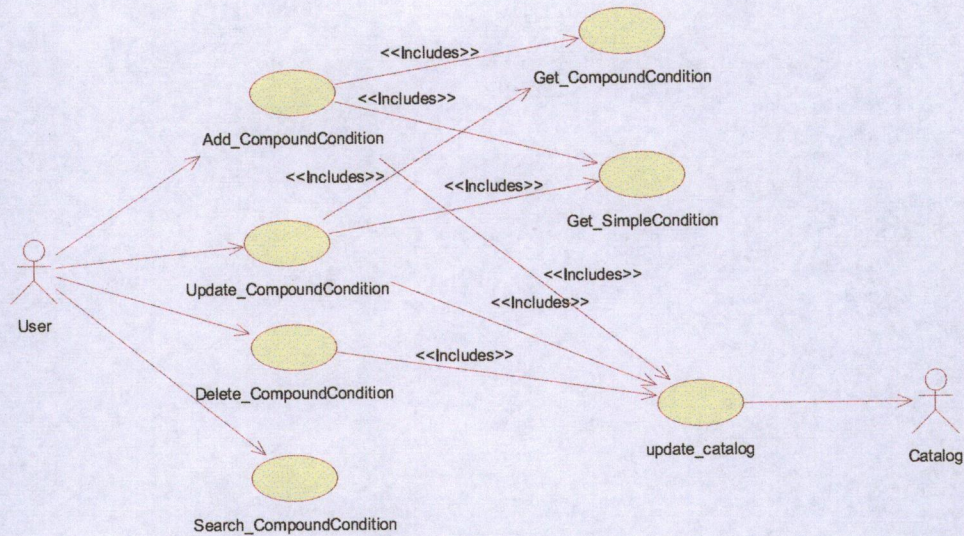


Figure 23: Compound Condition Use Case Diagram

Compound Condition Use Case Description:

Goal: User will be able to Add/Delete/Update/Search the Compound Condition.

Actors: Non-Specialist User, System Catalogue.

Pre-Conditions: User gets the refreshed page to add new actions. To delete, update and search a Compound Condition is already saved to the system catalogue.

Post-Condition: Compound Condition is Added/Updated/Deleted/Searched successfully.

5-4-2-Class Diagram:

Class diagrams represent the static view of the system. The structure which contains all system classes their attributes and relationships among these classes. Classes are UML notations to depict all entities which directly or indirectly are stake holders of the system. No dynamic information is represented using this representation.

The following below diagram shows the static structure of Policy Editor for SANTA. It shows all the system classes with corresponding attributes and operations. Class hierarchies are shows by using associations and inheritance relationships [13].

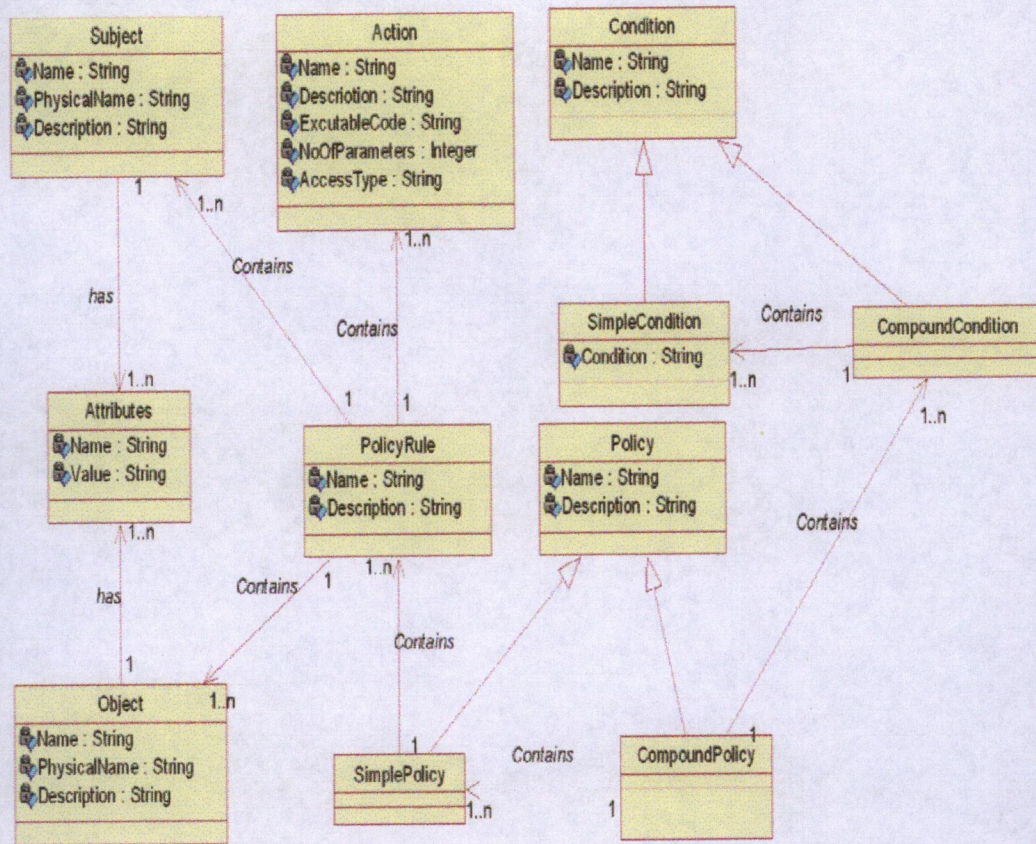


Figure 24: System Class Diagram

5-4-3-Sequence Diagram:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Every sequence diagram represents a use case scenario which depicts the entire communication and message passing sequence. Below are the sequence diagrams for Policy Editor for SANTA and every single diagram represents the individual use case scenario in the system [14].

Action:

Add Action:

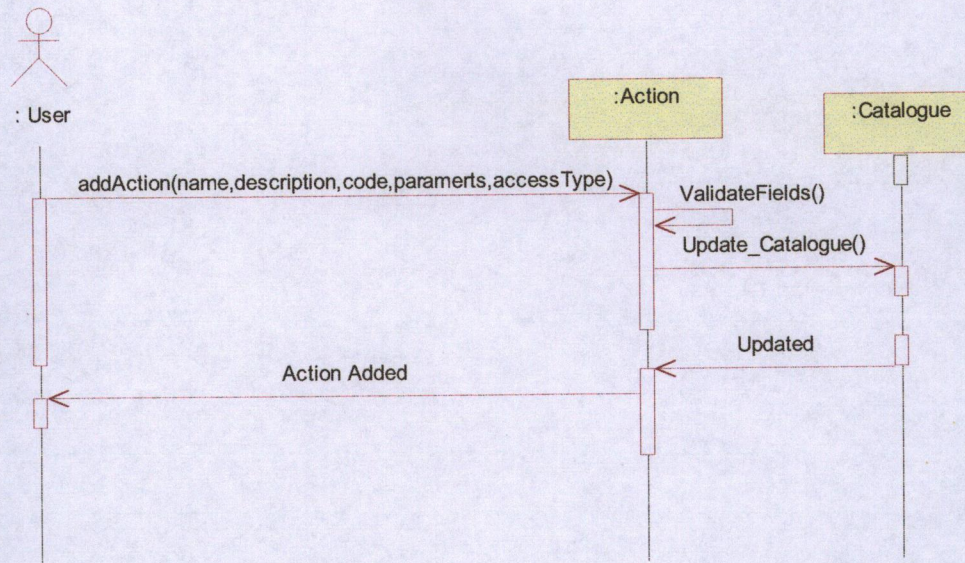


Figure 25: Add Action Diagram

User has to provide all the required parameters in order to add new action to the system. That will be performed in the class of Action which contains all the required functions and attributes to perform this action. Once all the fields are validated, Action will be saved to the database and user will be given an acknowledgement of successful addition

Delete Action:

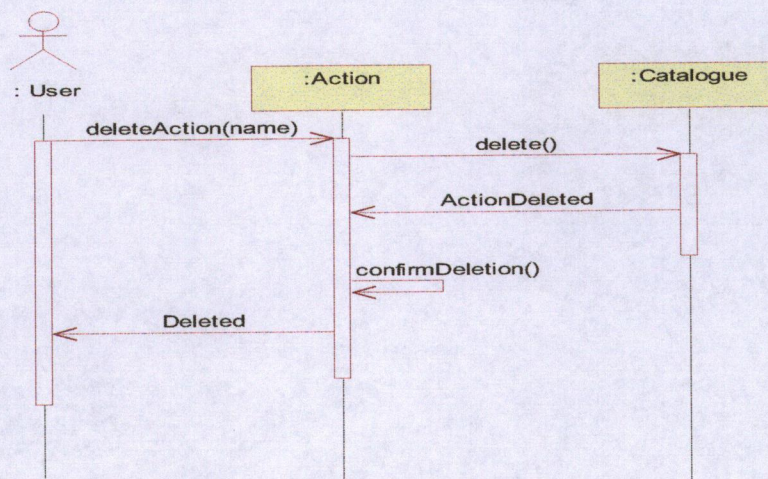


Figure 26: Delete Action Diagram

Deletion of any action will be made based on the provided name of the action. On successful deletion database would be updated and acknowledgement will be provided to the user.

Search Action:

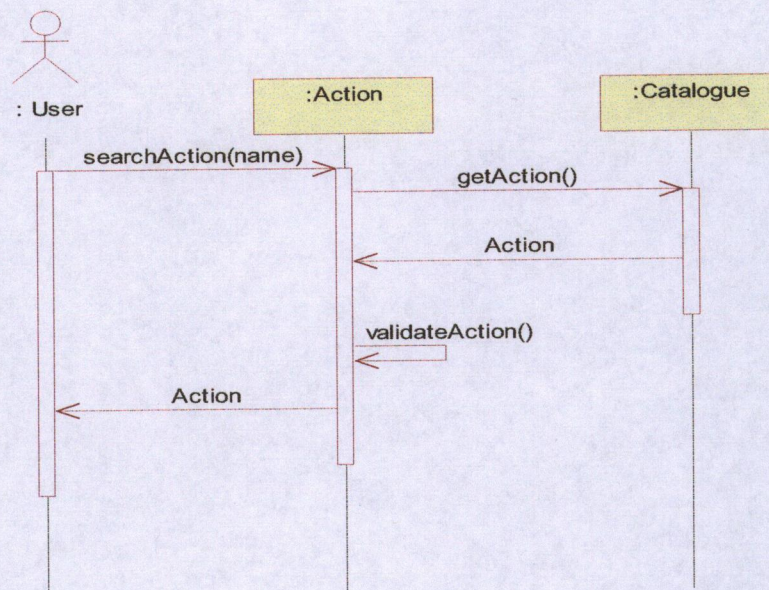


Figure 26: Search Action Diagram

In order to search any action, that action should be saved and user has to provide that action name to be searched. Once action searched, it will be shown to the user with the respective fields.

Update Action:

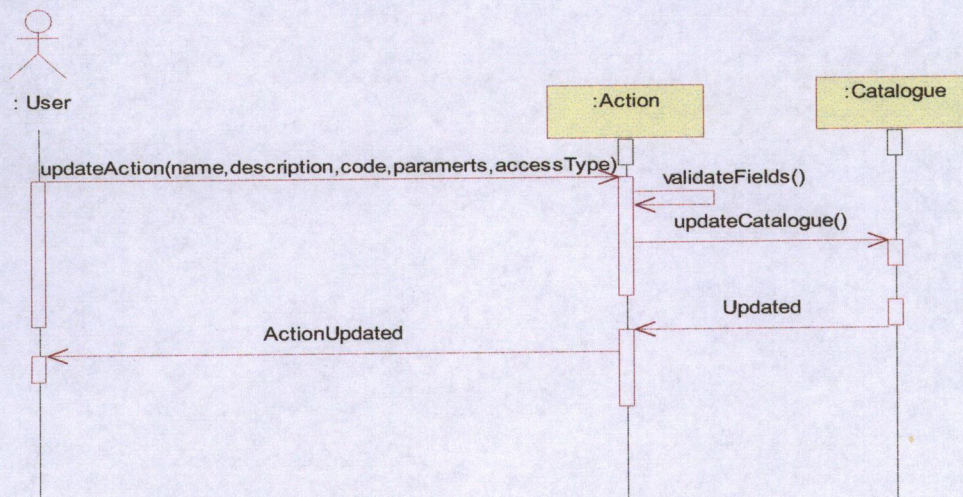


Figure 28: Update Action Diagram

In order to update any action which has been already added by the user, any field can be changed and provided to be updated. If user do not change any fields and still wants to update the action all the fields would remain same. If user changes any fields than all the changes will be written to database and next time user will see the changes to that action. On successful update user will be given a success acknowledgement.

Subject:

Add Subject:

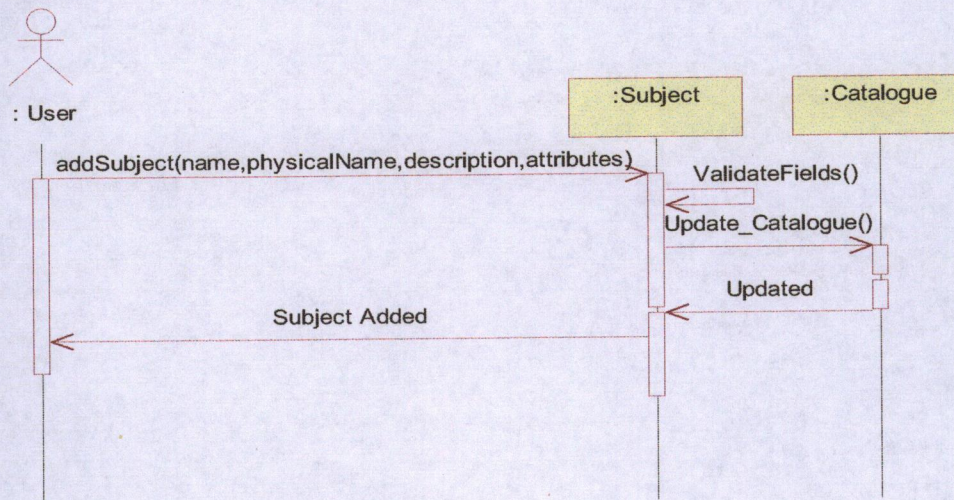


Figure 29: Add Subject Diagram

User has to provide all the required parameters in order to add new subject to the system. That will be performed in the class of Subject which contains all the required functions and attributes to perform this action. Once all the fields are validated, Subject will be saved to the database and user will be given an acknowledgement of successful addition

Delete Subject:

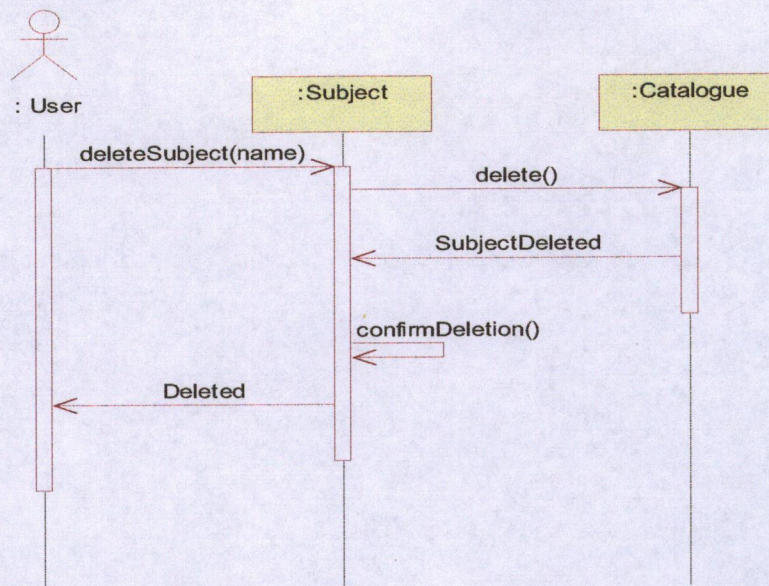


Figure 30: Delete Subject Diagram

Deletion of any subject will be made based on the provided name of the Subject. On successful deletion database would be updated and acknowledgement will be provided to the user.

Update Subject:

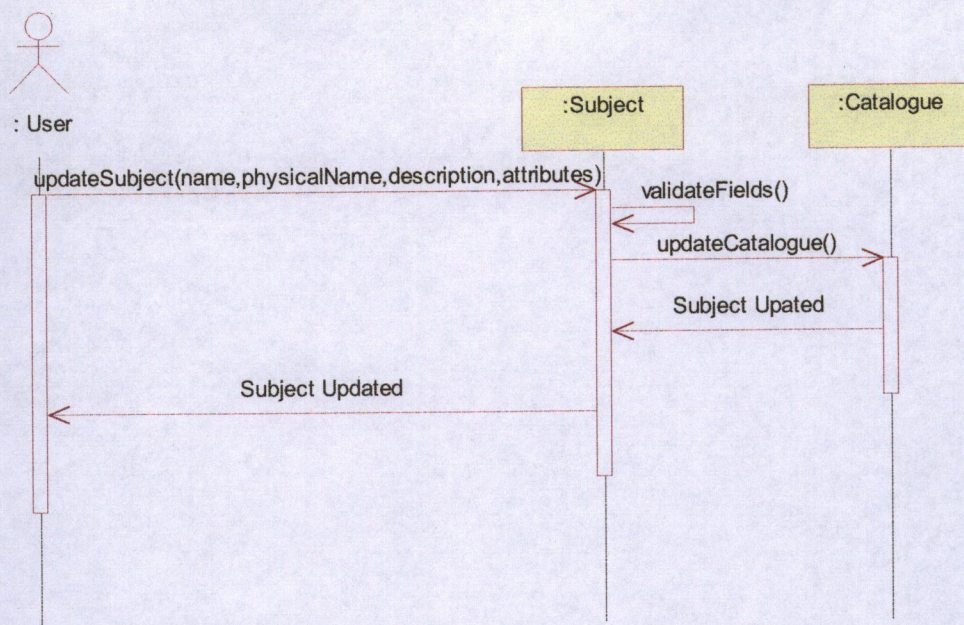


Figure 31: Update Subject Diagram

In order to update any subject which has been already added by the user, any field can be changed and provided to be updated. If user do not change any fields and still wants to update the subject all the fields would remain same. If user changes any fields than all the changes will be written to database and next time user will see the changes to that subject. On successful update user will be given a success acknowledgement.

Search Subject:

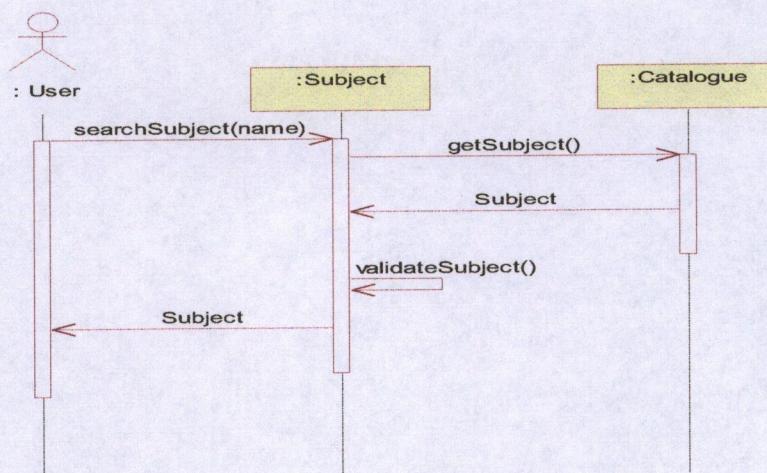


Figure 32: Search Subject Diagram

Subject would be searched based on the provided name. Fields would be set to the corresponding searched data and acknowledgement will be give to user on successful search.

Object:
Add Object:

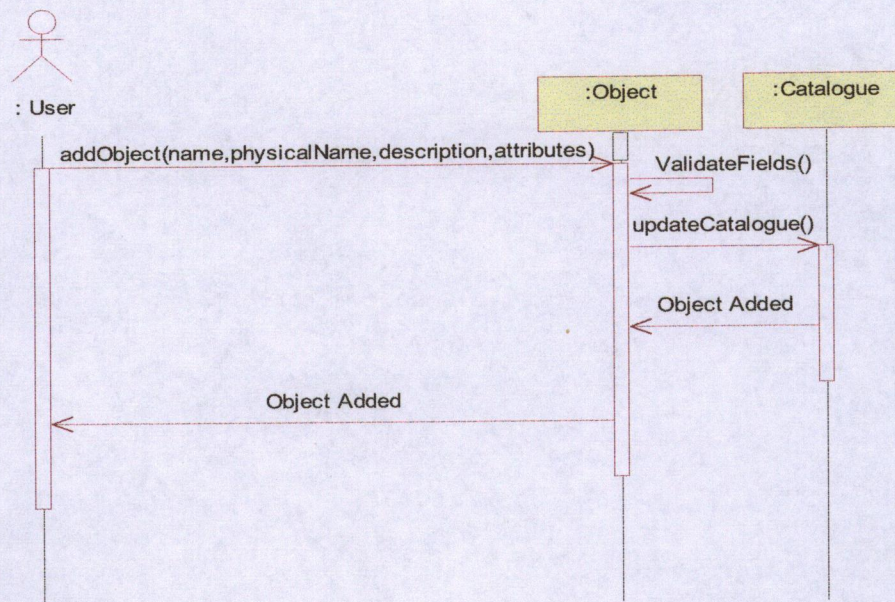


Figure 33: Add Object Diagram

User has to provide all the required parameters in order to add new Object to the system. That will be performed in the class of Object which contains all the required functions and attributes to perform this action. Once all the fields are validated, Object will be saved to the database and user will be given an acknowledgement of successful addition.

Delete Object:

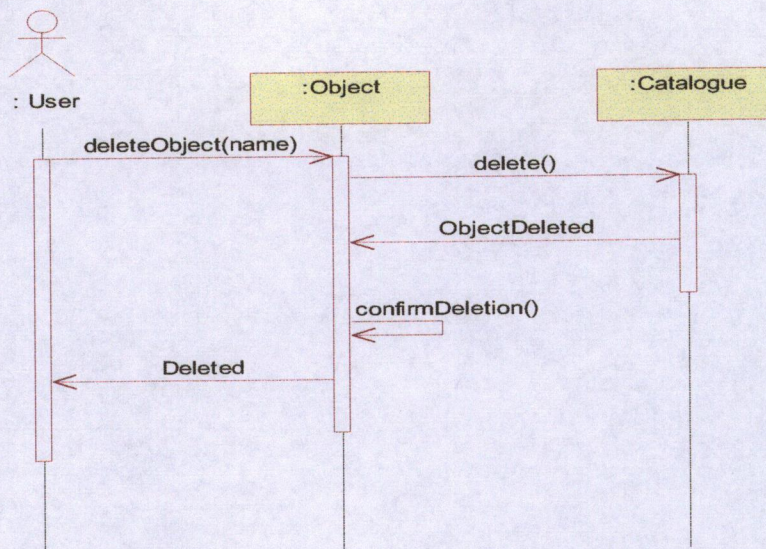


Figure 34: Delete Object Diagram

Deletion of any object will be made based on the provided name of the Object. On successful deletion database would be updated and acknowledgement will be provided to the user.

Update Object:

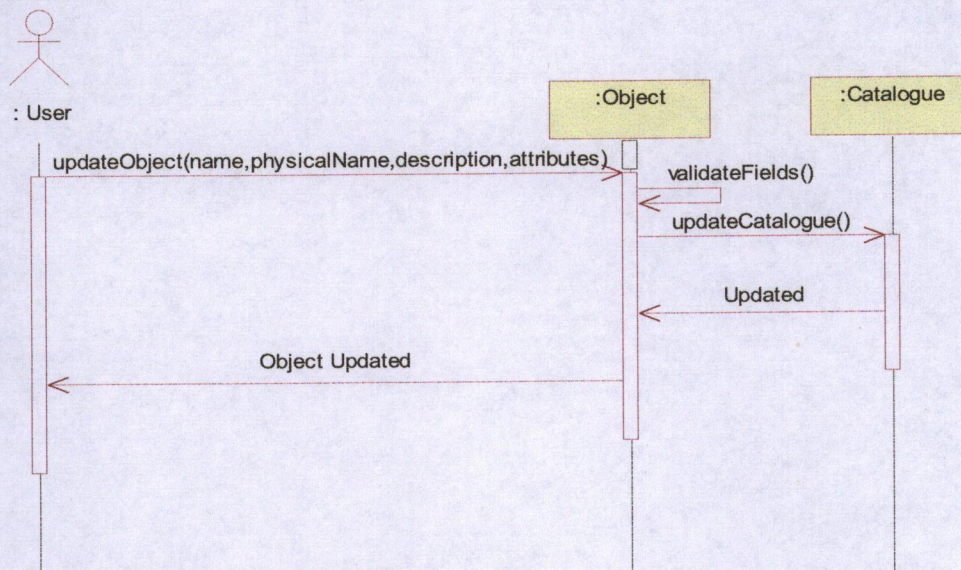


Figure 35: Update Object Diagram

In order to update any object which has been already added by the user, any field can be changed and provided to be updated. If user do not change any fields and still wants to update the object all the fields would remain same. If user changes any fields than all the changes will be written to database and next time user will see the changes to that action. On successful update user will be given a success acknowledgement.

Search Object:

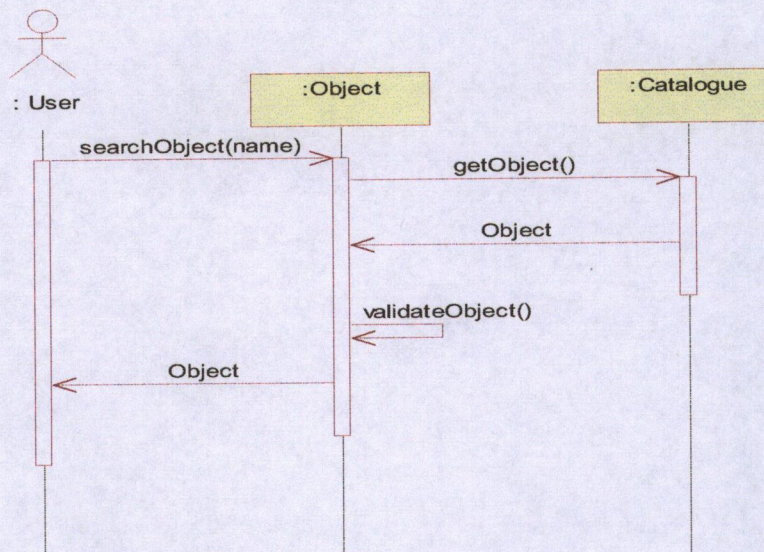


Figure 36: Search Object Diagram

Object would be searched based on the provided name. Fields would be set to the corresponding searched data and acknowledgement will be give to user on successful search.

Policy Rule:

Add Policy Rule:

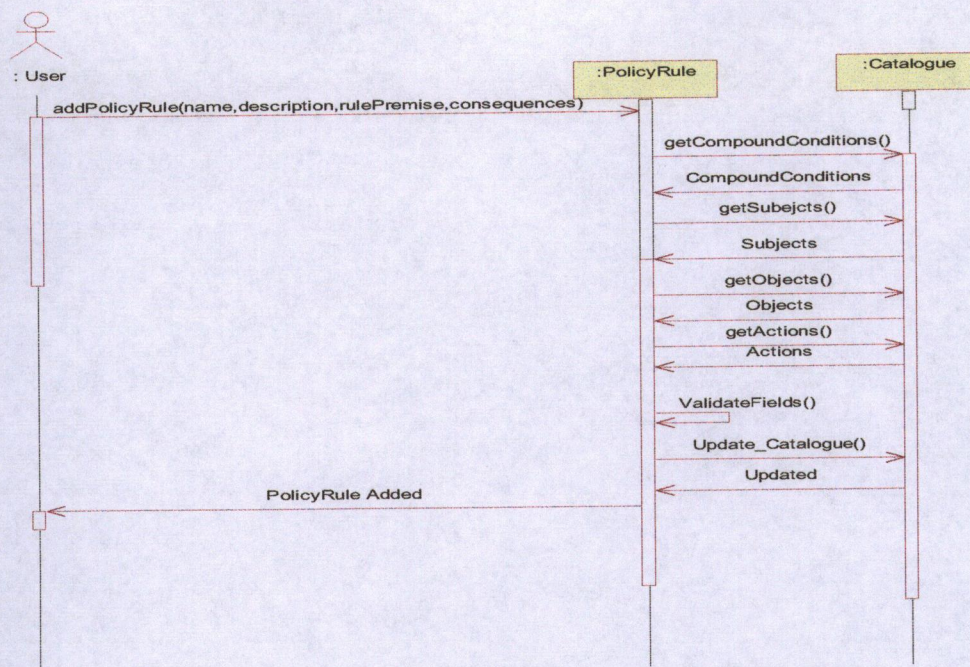


Figure 37: Add Policy Rule Diagram

User has to provide all the required parameters in order to add new Policy Rule to the system. That will be performed in the class of Policy Rule which contains all the required functions and attributes to perform this action. Consequences would be selected by user by selecting all the available Subjects, Objects and Actions. Once all the fields are validated, Policy Rule will be saved to the database and user will be given an acknowledgement of successful addition.

Delete Policy Rule:

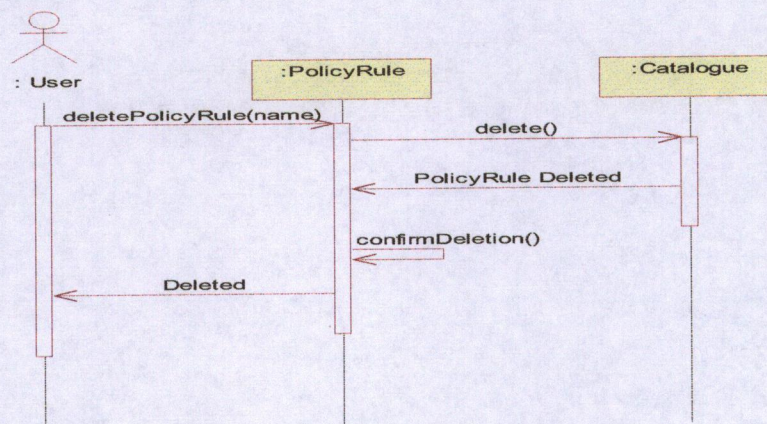


Figure 38: Delete Object Diagram

Deletion of any Policy Rule will be made based on the provided name of the Policy Rule. On successful deletion database would be updated and acknowledgement will be provided to the user

Update Policy Rule:

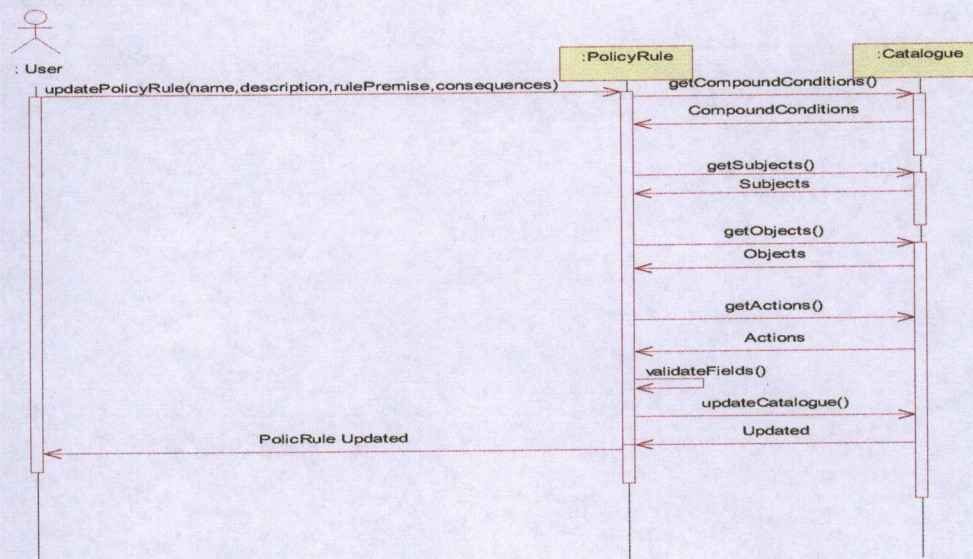


Figure 39: Update Policy Rule Diagram

In order to update any Policy Rule which has been already added by the user, any field can be changed and provided to be updated. If user do not change any fields and still wants to update the Policy Rule all the fields would remain same. If user changes any fields than all the changes will be written to database and next time user will see the changes to that Policy Rule. On successful update user will be given a success acknowledgement.

Search Policy Rule:

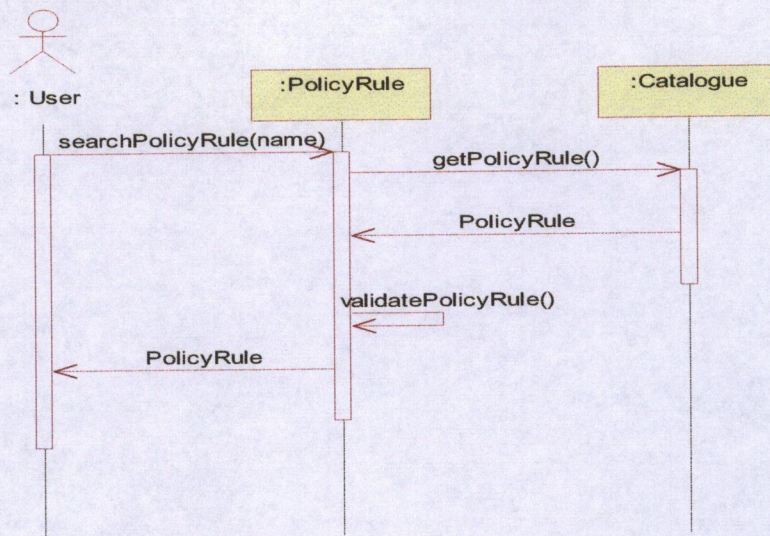


Figure 40: Search Policy Rule Diagram

Policy Rule would be searched based on the provided name. All the saved compound, simple policies and actions will also be retrieved with the specific searched Compound Policy. Fields would be set to the corresponding searched data and acknowledgement will be give to user on successful search.

**Simple Policy:
Add Simple Policy:**

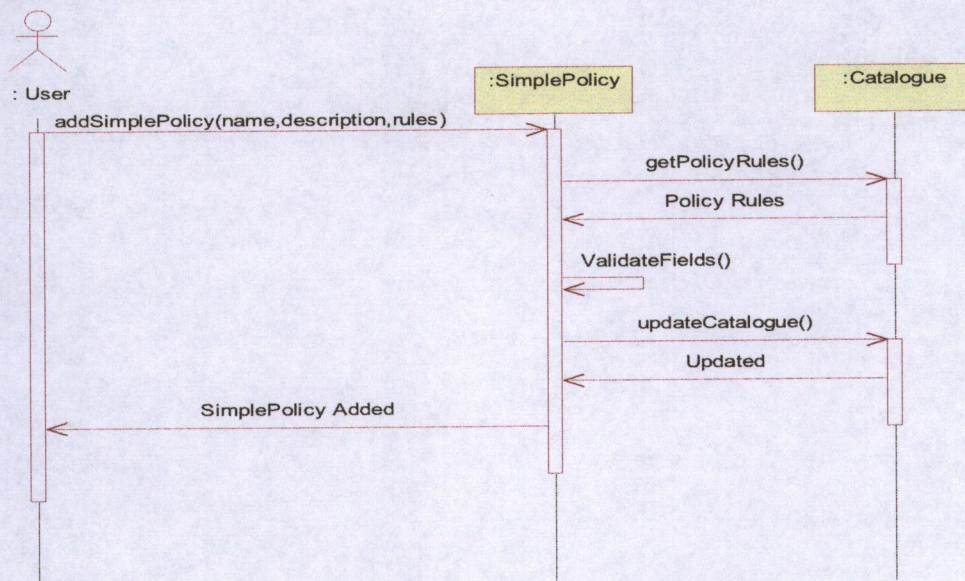


Figure 41: Add Simple Policy Diagram

User has to provide all the required parameters in order to add new Simple Policy to the system. That will be performed in the class of Simple Policy which contains all the required functions and attributes to perform this action. Rules will be retrieved from database. These are rules which already added by the users. Once all the fields are validated, Simple Policy will be saved to the database and user will be given an acknowledgement of successful addition.

Update Simple Policy:

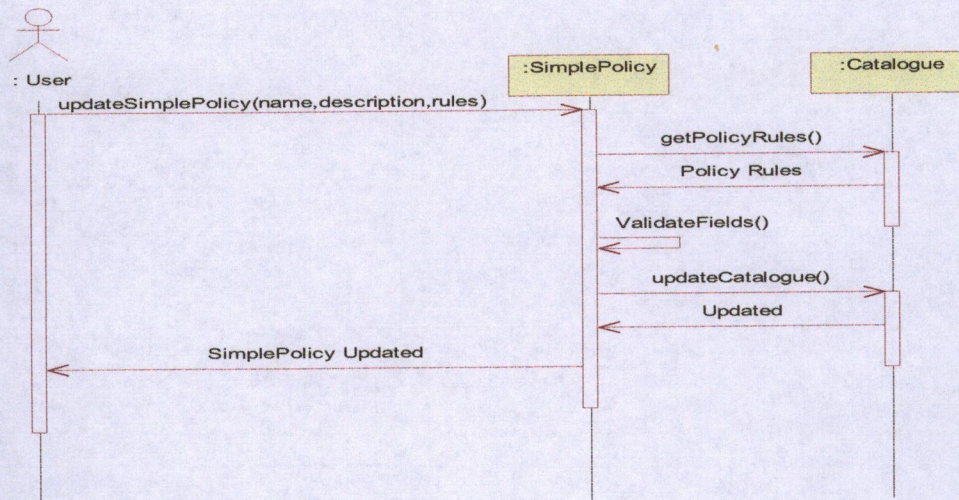


Figure 42: Update Simple Policy Diagram

In order to update any Simple Policy which has been already added by the user, any field can be changed and provided to be updated. If user do not change any fields and still wants to update the Simple Policy all the fields would remain same. If user changes any fields than all the changes will be written to database and next time user will see the changes to that Simple Policy. On successful update user will be given a success acknowledgement.

Delete Simple Policy:

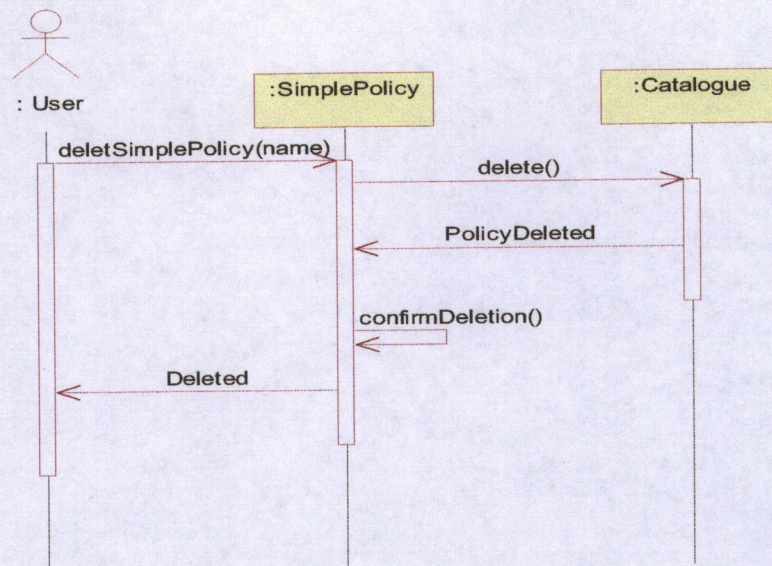


Figure 43: Delete Simple Policy Diagram

Deletion of any Simple Policy will be made based on the provided name of the Simple Policy. On successful deletion database would be updated and acknowledgement will be provided to the user.

Search Simple Policy:

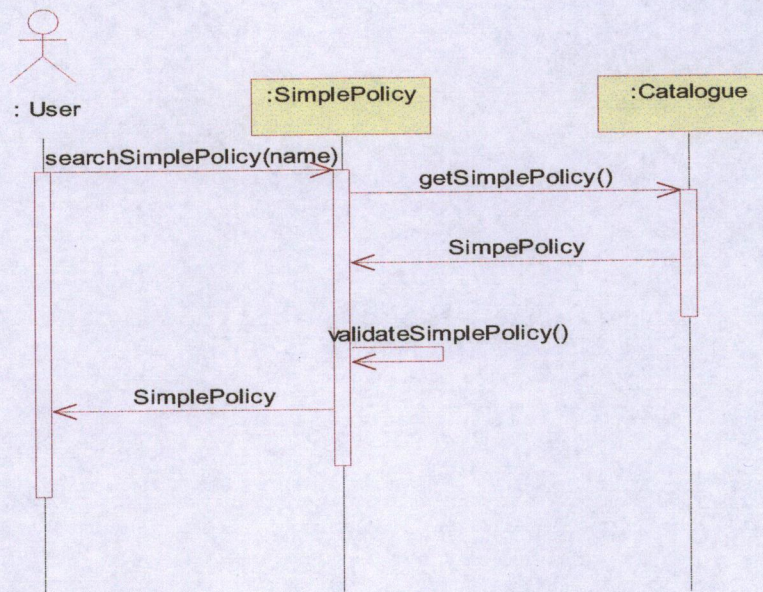


Figure 44: Search Policy Rule Diagram

Simple Policy would be searched based on the provided name. All the saved Policy Rules will also be retrieved with the specific searched simple policy. Fields would be set to the corresponding searched data and acknowledgement will be give to user on successful search.

**Compound Policy1:
Add Compound Policy1:**

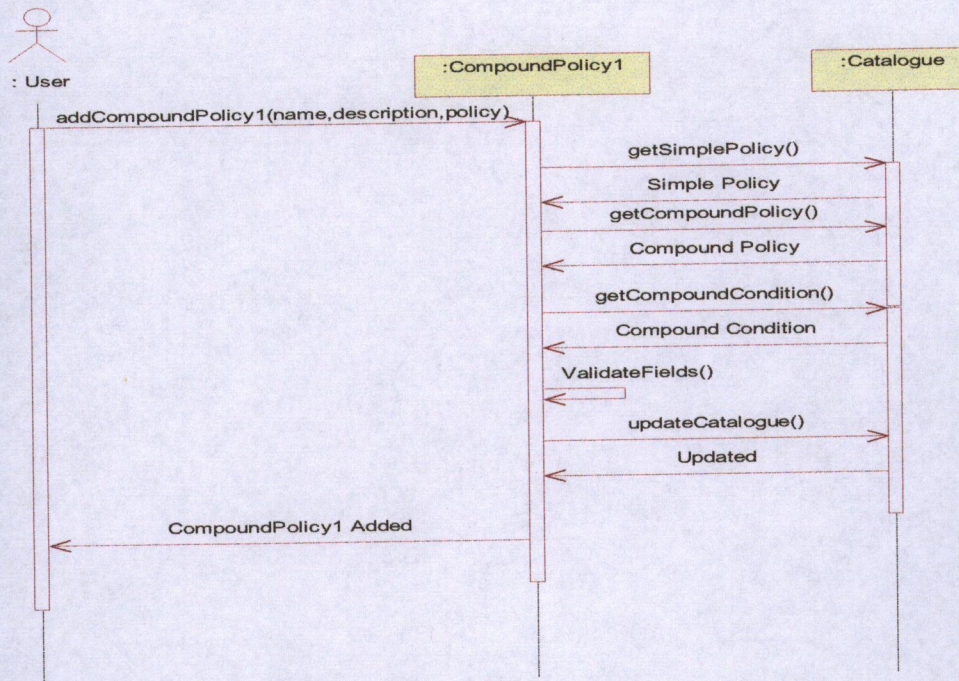


Figure 45: Add Compound Policy1 Diagram

User has to provide all the required parameters in order to add new Compound Policy 1 to the system. That will be performed in the class of Compound Policy 1 which contains all the required functions and attributes to perform this action. Simple Policies and Compound Policies will be retrieved from database to be selected by user to save a new compound policy 1. Once all the fields are validated, Compound Policy 1 will be saved to the database and user will be given an acknowledgement of successful addition.

Delete Compound Policy1:

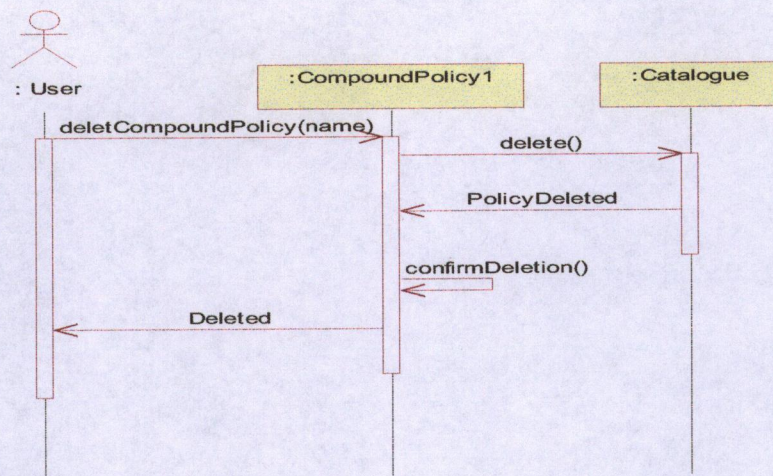


Figure 46: Delete Compound Policy1 Diagram

Deletion of any Compound Policy 1 will be made based on the provided name of the Compound Policy 1. On successful deletion database would be updated and acknowledgement will be provided to the user.

Update Compound Policy1:

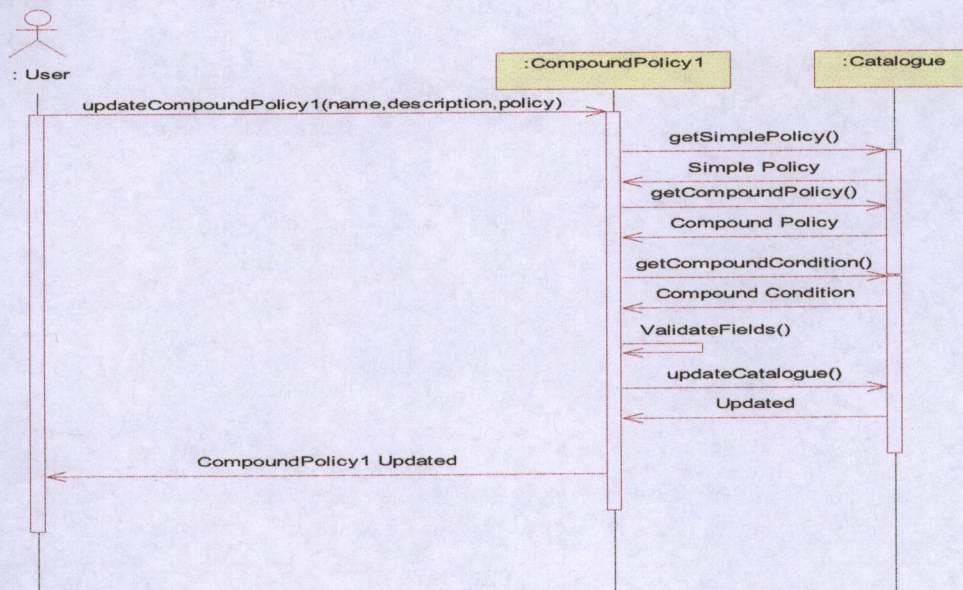


Figure 47: Update Compound Policy1 Diagram

In order to update any Compound Policy 1 which has been already added by the user, any field can be changed and provided to be updated. If user do not change any fields and still wants to update the Compound Policy 1 all the fields would remain same. If

user changes any fields than all the changes will be written to database and next time user will see the changes to that Compound Policy 1. On successful update user will be given a success acknowledgement.

Search Compound Policy1:

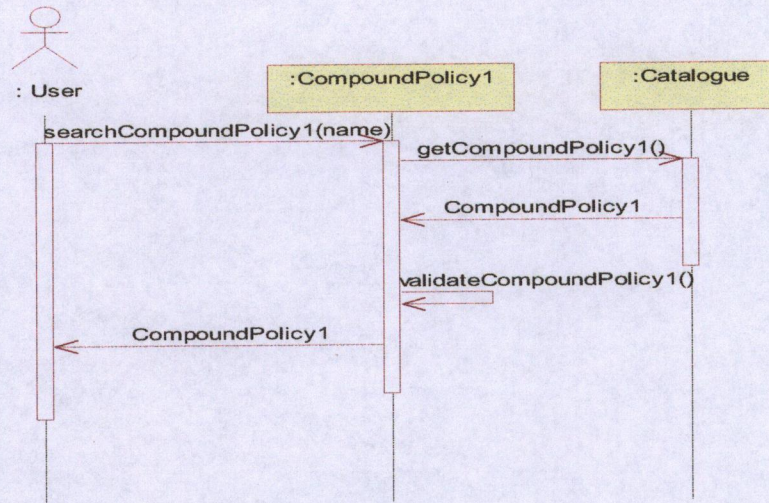


Figure 48: Search Compound Policy1 Diagram

Compound Policy 1 would be searched based on the provided name. All the saved compound and simple policies will also be retrieved with the specific searched compound policy 1. Fields would be set to the corresponding searched data and acknowledgement will be give to user on successful search.

Compound Policy2: Add Compound Policy2:

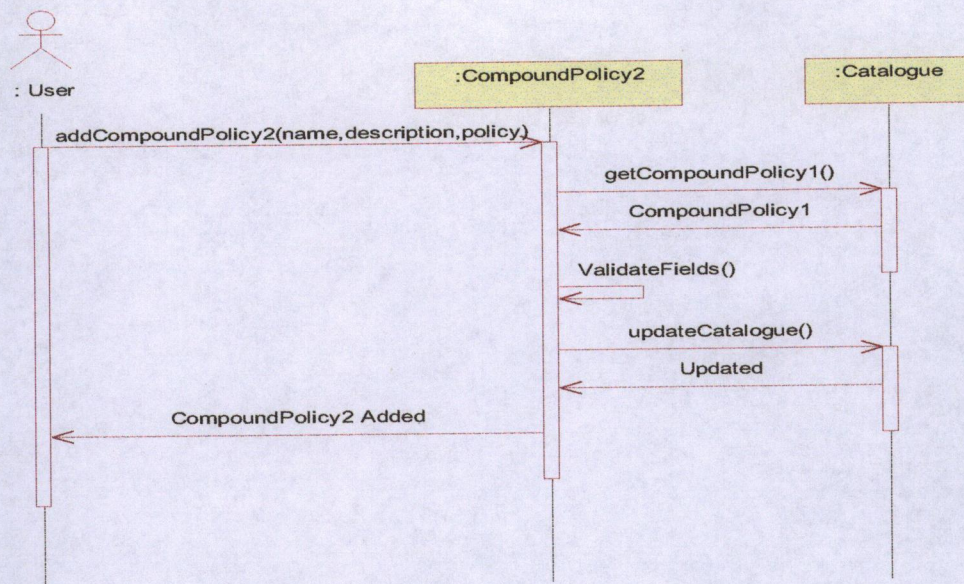


Figure 49: Add Compound Policy2 Diagram

User has to provide all the required parameters in order to add new Compound Policy 2 to the system. That will be performed in the class of Compound Policy 2 which contains all the required functions and attributes to perform this action. Simple Policies and Compound Policies will be retrieved from database to be selected by user to save a new compound policy 2. Once all the fields are validated, Compound Policy 2 will be saved to the database and user will be given an acknowledgement of successful addition.

Delete Compound Policy2:

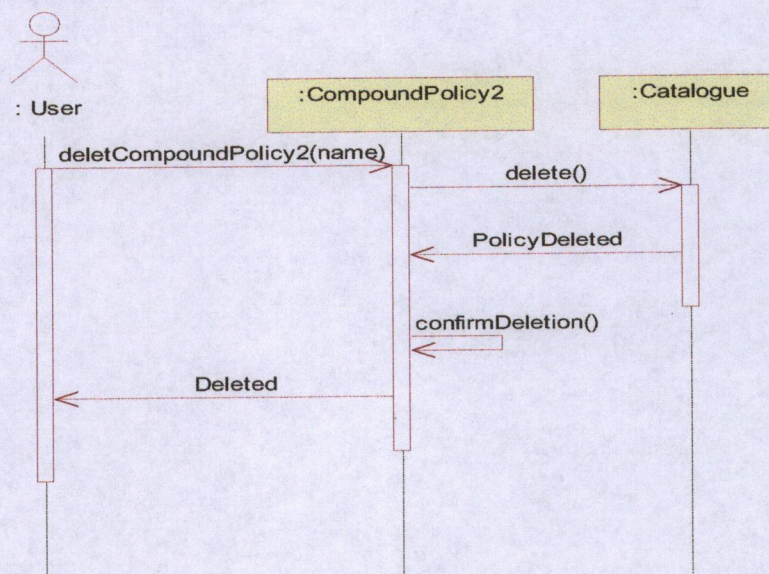


Figure 50: Delete Compound Policy2 Diagram

Deletion of any Compound Policy 2 will be made based on the provided name of the Compound Policy 2. On successful deletion database would be updated and acknowledgement will be provided to the user.

Update Compound Policy2:

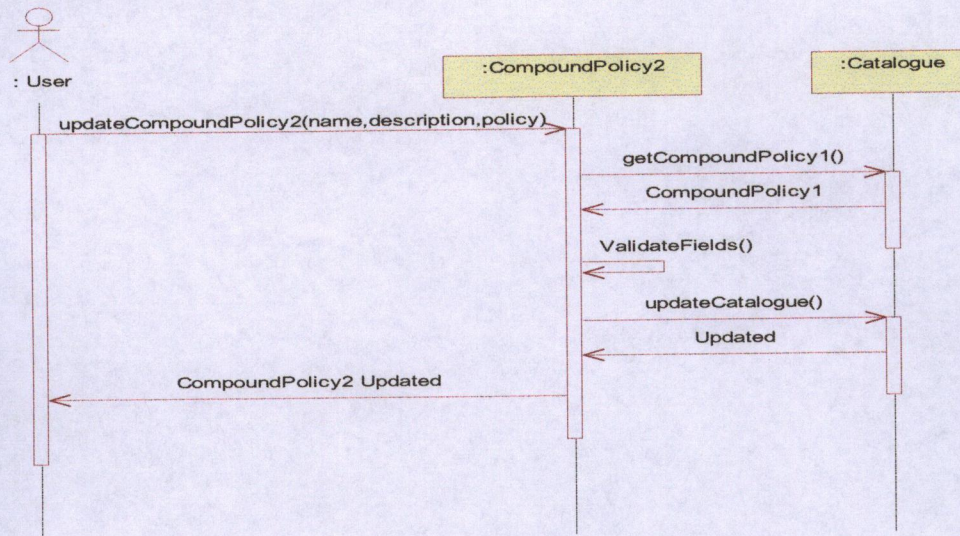


Figure 51: Update Compound Policy2 Diagram

In order to update any Compound Policy 2 which has been already added by the user, any field can be changed and provided to be updated. If user do not change any fields and still wants to update the Compound Policy 2 all the fields would remain same. If user changes any fields than all the changes will be written to database and next time user will see the changes to that Compound Policy 2. On successful update user will be given a success acknowledgement.

Search Compound Policy2:

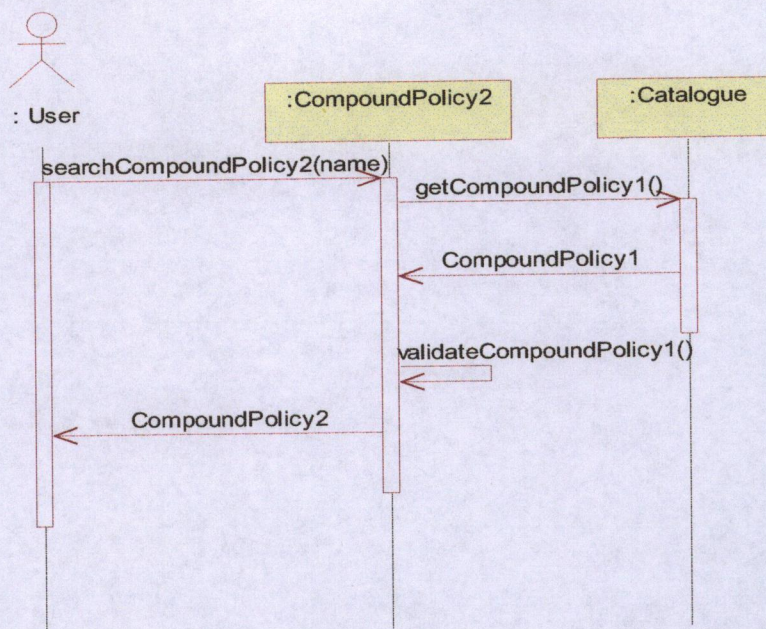


Figure 52: Search Compound Policy2 Diagram

Compound Policy 2 would be searched based on the provided name. All the saved compound and simple policies will also be retrieved with the specific searched compound policy 2. Fields would be set to the corresponding searched data and acknowledgement will be give to user on successful search.

Simple Condition:

Add Simple Condition:

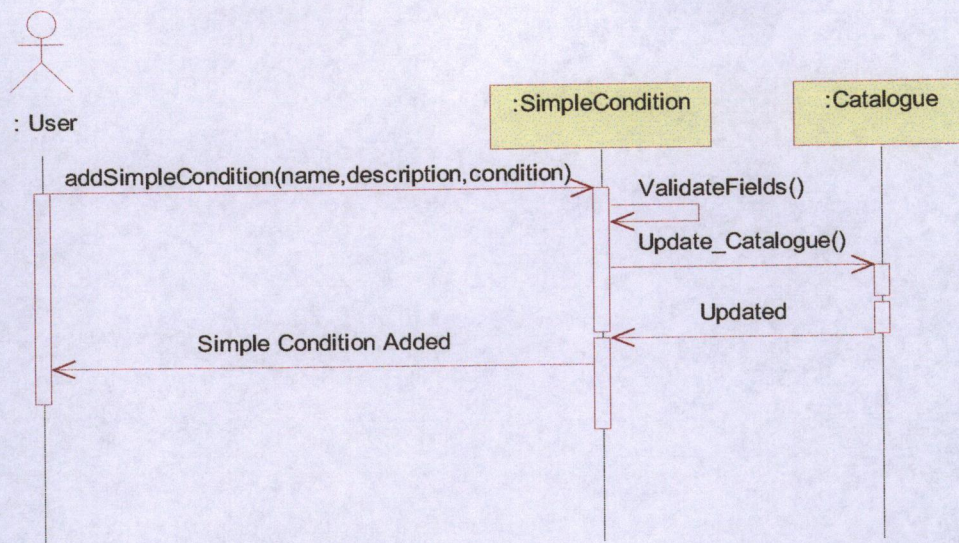


Figure 53: Add Simple Condition Diagram

User has to provide all the required parameters in order to add new Simple Condition to the system. That will be performed in the class of Simple Condition which contains all the required functions and attributes to perform this action. Once all the fields are validated, Simple Condition will be saved to the database and user will be given an acknowledgement of successful addition.

Delete Simple Condition:

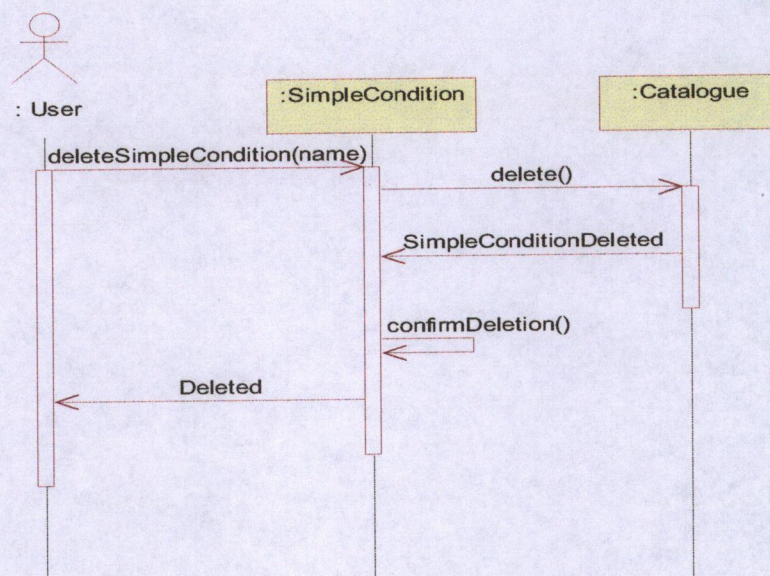


Figure 54: Delete Simple Condition Diagram

Update Simple Condition:

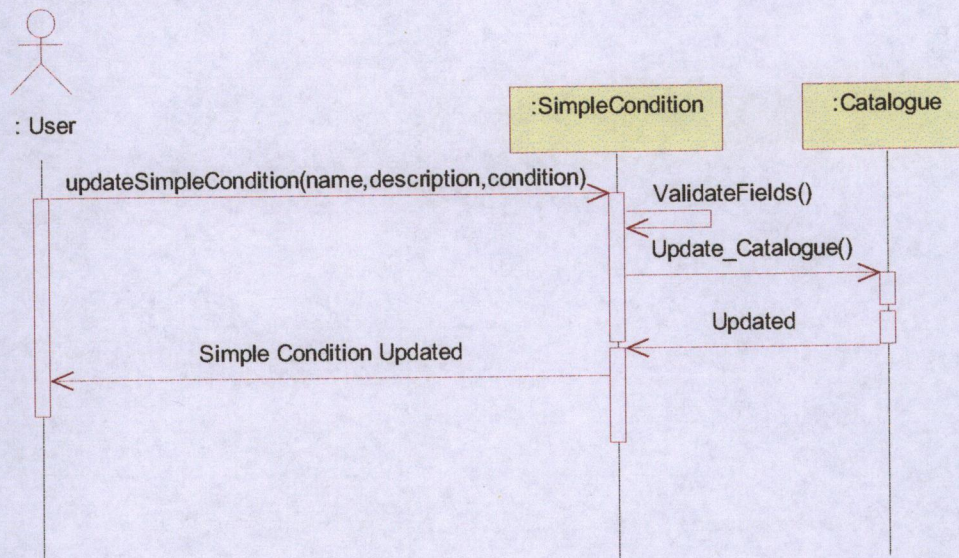


Figure 55: Update Simple Condition Diagram

In order to update any Simple Condition which has been already added by the user, any field can be changed and provided to be updated. If user do not change any fields and still wants to update the Simple Condition all the fields would remain same. If user changes any fields than all the changes will be written to database and next time user will see the changes to that Simple Condition. On successful update user will be given a success acknowledgement.

Search Simple Condition:

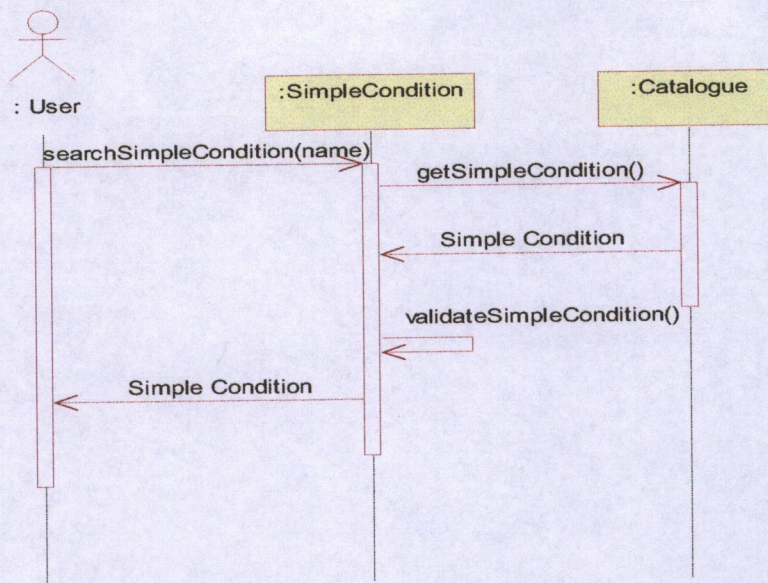


Figure 56: Search Simple Condition Diagram

Simple Condition would be searched based on the provided name. Fields would be set to the corresponding searched data and acknowledgement will be give to user on successful search.

Compound Condition:
Add Compound Condition:

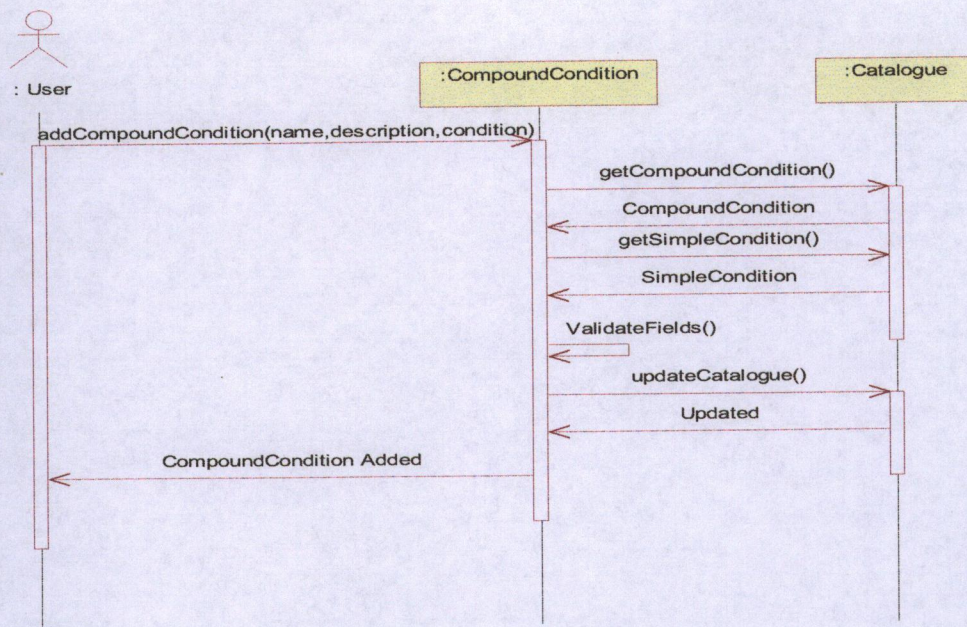


Figure 57: Add Compound Condition Diagram

To add a new compound condition, all the fields provided with name. Before doing this action all the saved compound conditions and simple condition will be retrieved from database so that use can select anyone of them. Once all the fields are provided, compound condition would be added to database and successful addition is acknowledged to the user.

Delete Compound Condition:

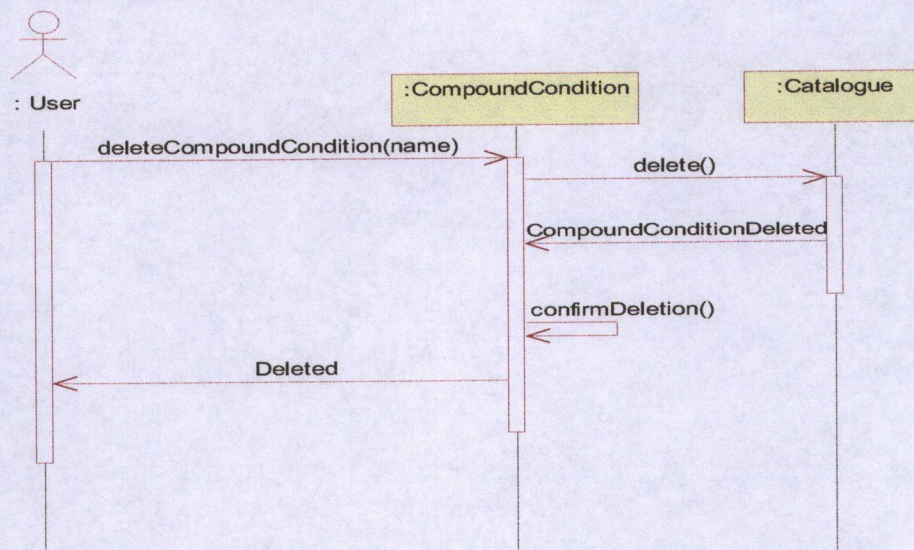


Figure 58: Delete Compound Condition Diagram

Deletion of compound condition would be made based on provided name of compound condition. This is the function of the class Compound Condition. All the data would be updated in the database as well and acknowledgement would also be give to the user on successful deletion.

Update Compound Condition:

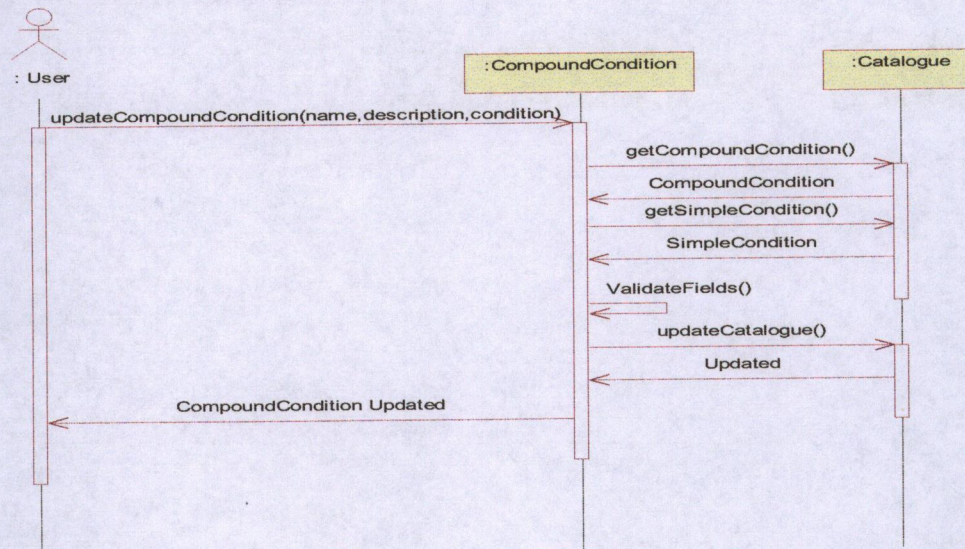


Figure 59: Update Compound Condition Diagram

In order to update any Compound Condition which has been already added by the user, any field can be changed and provided to be updated. If user do not change any fields and still wants to update the Compound Condition all the fields would remain same. If user changes any fields than all the changes will be written to database and next time user will see the changes to that Compound Condition. On successful update user will be given a success acknowledgement.

Search Compound Condition:

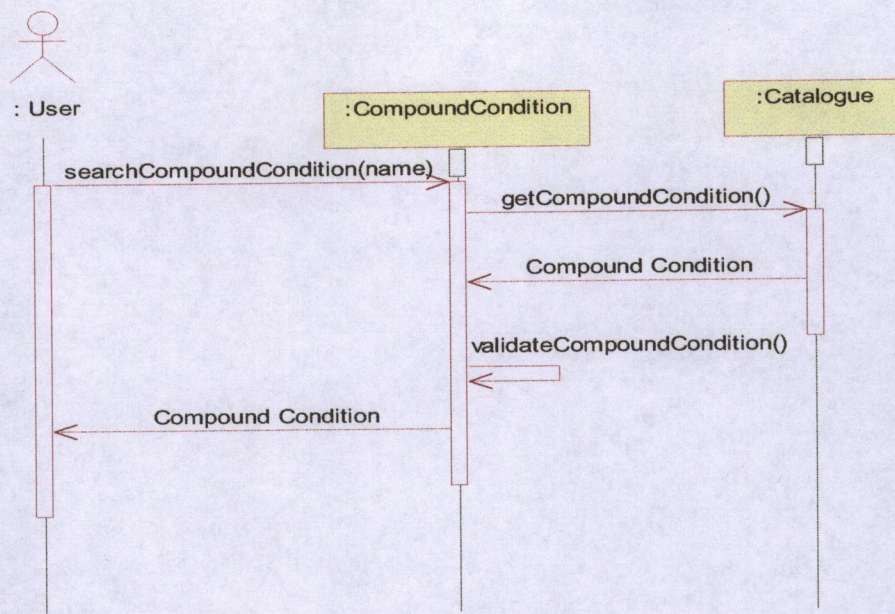


Figure 60: Search Compound Condition Diagram

Compound condition would be searched based on the provided name. All the saved compound conditions will also be retrieved with the specific searched compound condition. Fields would be set to the corresponding searched data and acknowledgement will be give to user on successful search.

5-5-Flow Char of Policy Editor for SANTA:

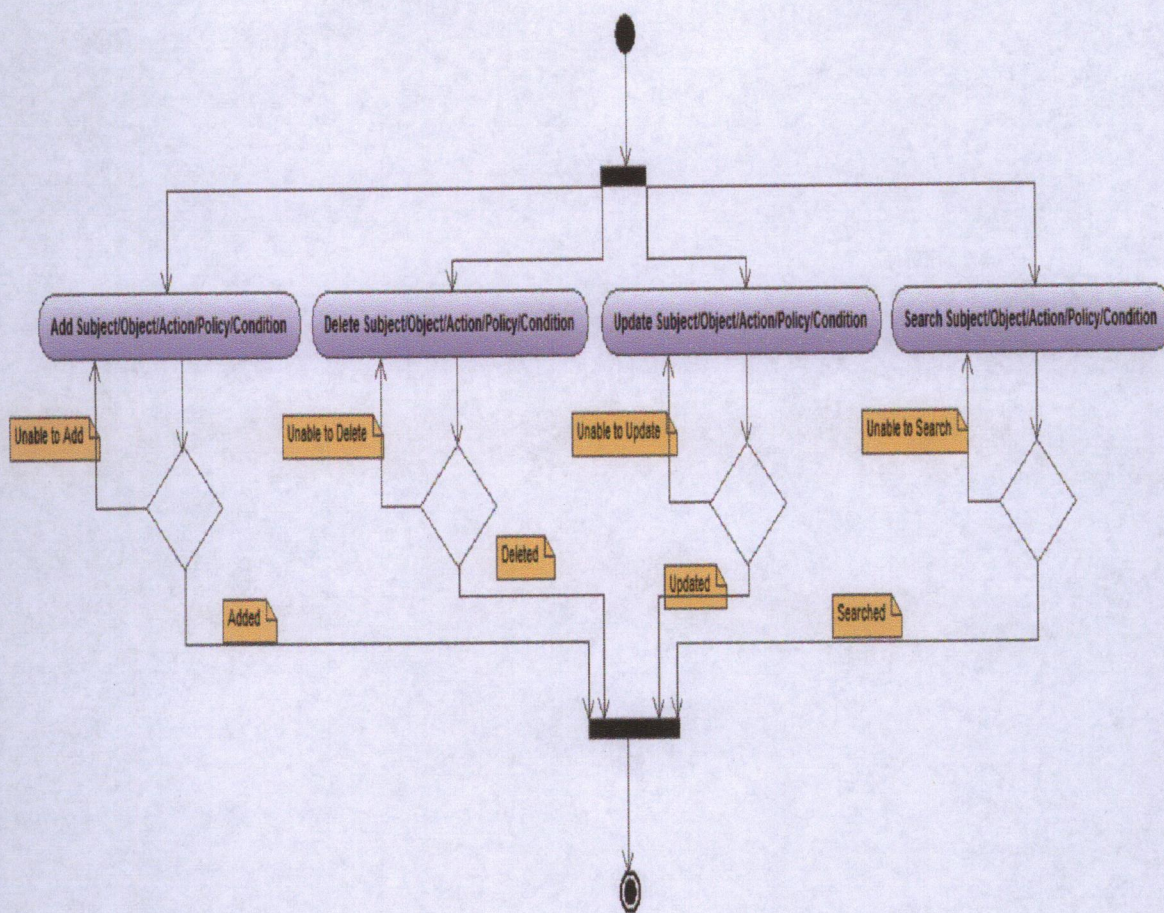


Figure 61: Flow chart for the system

User will have all the options at start to perform any activity for security policy. Flow chart above demonstrates the entire flow of the system and exits only on successful activity. The system enables the user to start with any activity regarding subjects, objects, actions, policies and conditions.

5-6-Summary

In this chapter, we analyzed and designed Policy Editor. It was analyzed by using UML notation. The analysis of the system requirements is very important to understand what the system needs and how it works. The system has 9 interfaces and each one has different function. So we designed these interfaces to be implemented as specified.

A web-based policy editor for SANTA	العنوان:
Al Mutairi, Abd Algder	المؤلف الرئيسي:
Siewe, Francois(Supervisor)	مؤلفين آخرين:
2009	التاريخ الميلادي:
Leicester	موقع:
1 - 105	الصفحات:
687206	رقم MD:
رسائل جامعية	نوع المحتوى:
English	اللغة:
رسالة ماجستير	الدرجة العلمية:
De Montfort University	الجامعة:
Faculty of Computing Sciences and Engineering	الكلية:
بريطانيا	الدولة:
Dissertations	قواعد المعلومات:
الأمن المعلوماتي، هندسة البرمجيات، الإنترنت، لغة البرامج	مواضيع:
https://search.mandumah.com/Record/687206	رابط:

Chapter 6: Implementation

6-1-Overview:

This chapter will describe the implementation details by demonstrating the system screenshots and description of how to use the system and tools. This chapter illustrates the main system requirements for the Policy Editor. All the screen shots of system will represents the GUI of the system.

6-2-System Requirements [Client Side]:

Client can be on any platform with the following installations.

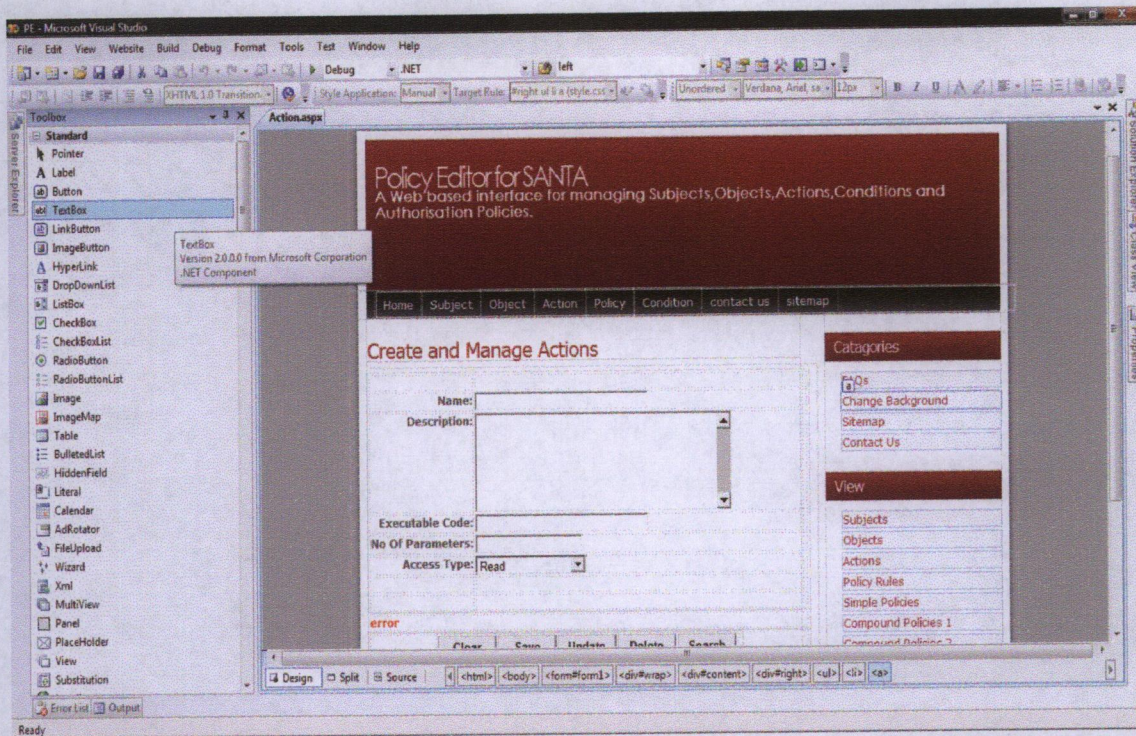
- Any Web Browser
- Internet Information Server (IIS).

Architectural Overview of System:

Policy Editor for SANTA has been developed under the architectural environment and 3-Tier architecture has been implemented. System has been divided into three main layers which are independent of each other. If any change made at any level it will not affect any other level of architecture. Tiers are implemented in ASP.NET and Ms SQL Server. Design tier is developed using the MS Visual Studio 2008 and basic GUI is designed using the dedicated and provided controls of ASP.NET. GUI is the first tier of the system to which user has to interact in order to use the system. Second tier is business logic which is the core to the 3-Tier of system. Business logic is developed using C# as a language which is called Code Behind in .Net. Business logic is directly associated with GUI and is integrated to connect to the DB layer of the system. The third and last layer of the system is Database layer which contains the database used in the system. .Net provides the libraries to integrate the application with database at abstract level. Developer just uses these libraries for database integration.

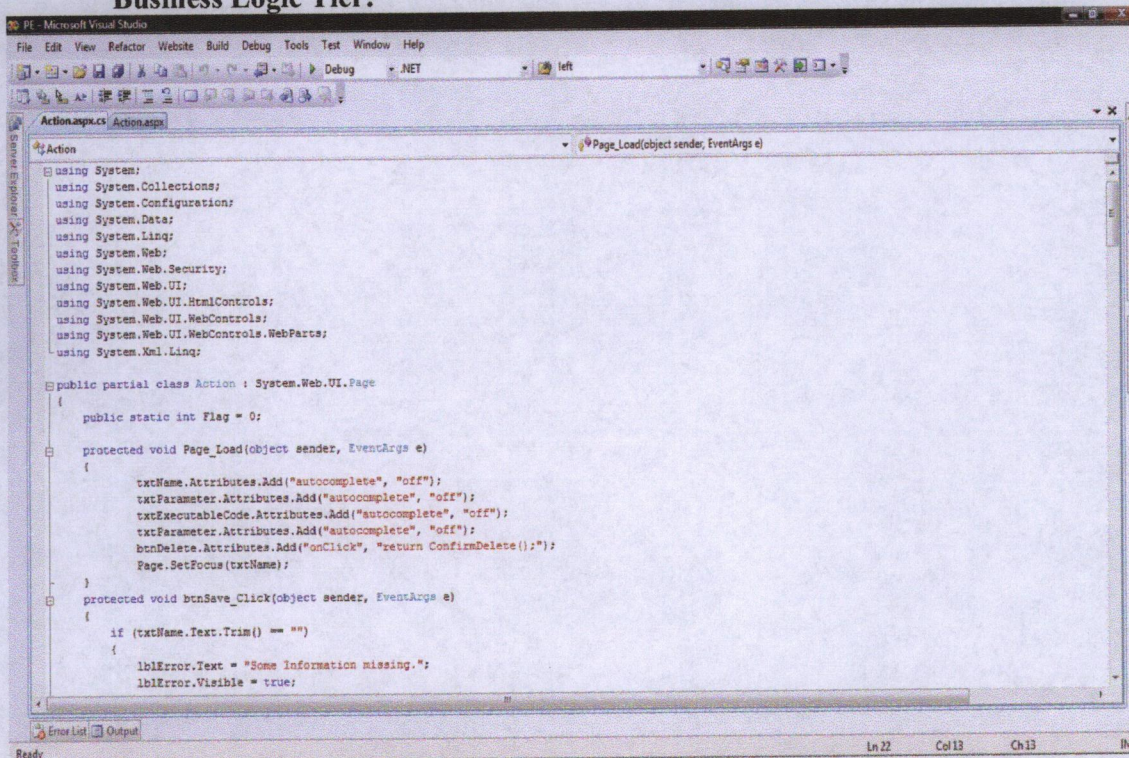
We will discuss our 3-Tier with a Sample Problem demonstration which will be the part of our developed system. For example, lest discuss the Action section of our system...

GUI Tier:



The screenshot above shows the main designs view of the Action page where user has been provided with the set of controls to use. On the left pane, we selected text box to be used in our page which can use used by drag and drop. So, this is the main idea how to use the controls during the design of any webpage. This is how all the pages have been designed for Policy Editor. This is the GUI Tier of the System from the architectural point of view.

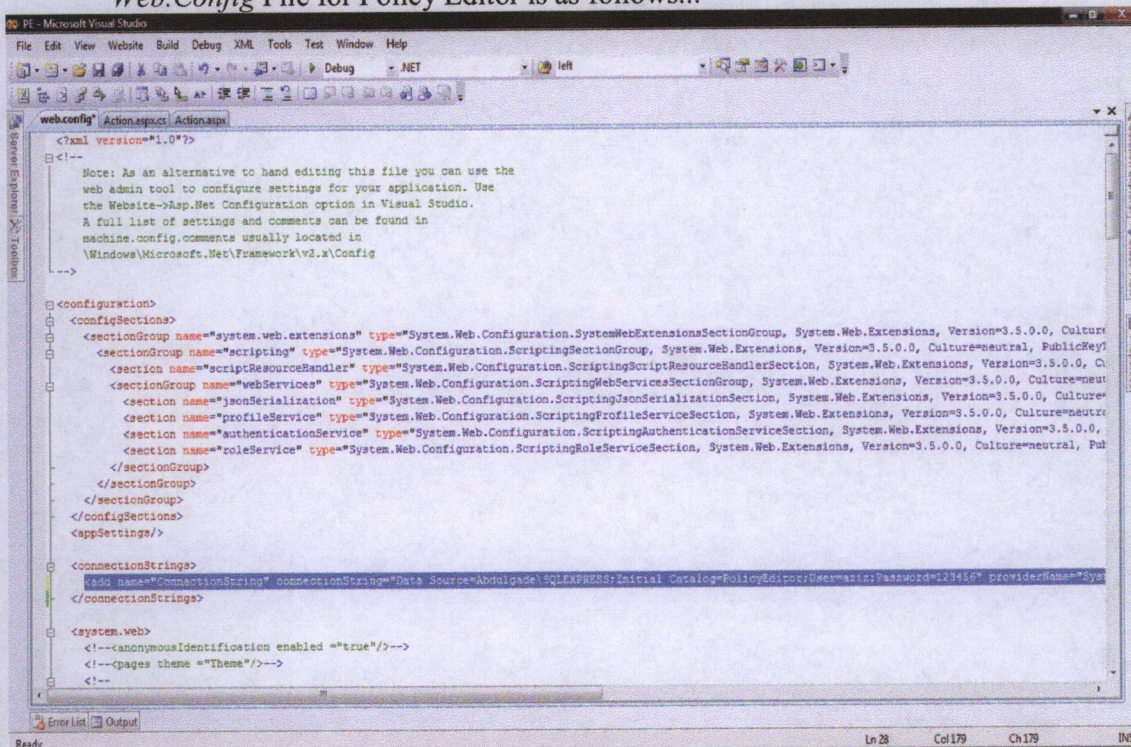
Business Logic Tier:



The screenshot above shows the main code behind file for the Action page. This is where main business logic for Action has been developed. On the top all the required

libraries has been added and some of the libraries are basic which are provided by default by .Net. For example, in order to use the database related components we have to add the library *System.Data* and it contains the *System.Data.SqlClient* component for database connectivity. In the code behind file we connect to the classes which provide the functionality of connecting to the database. We are using the *Web.Config* which contains all the configuration setting. Most frequently used configuration in our system is database connection which has been provided in using the Connection String and that Connection String is used in the business logic for data integration.

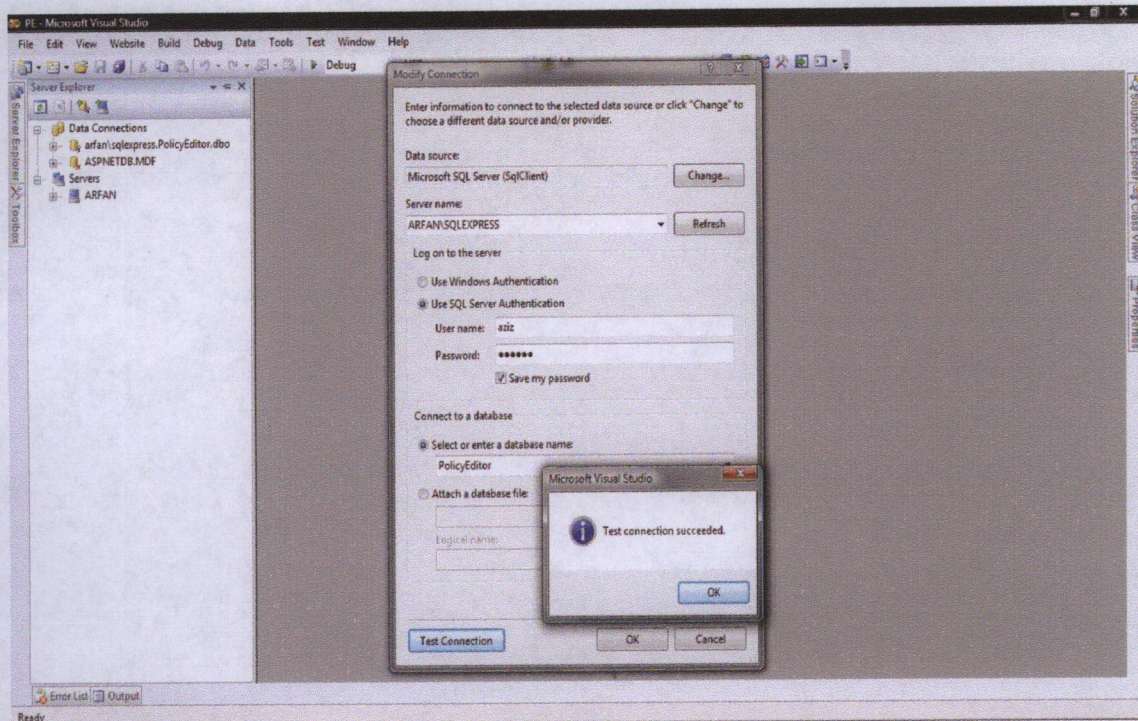
Web.Config File for Policy Editor is as follows...



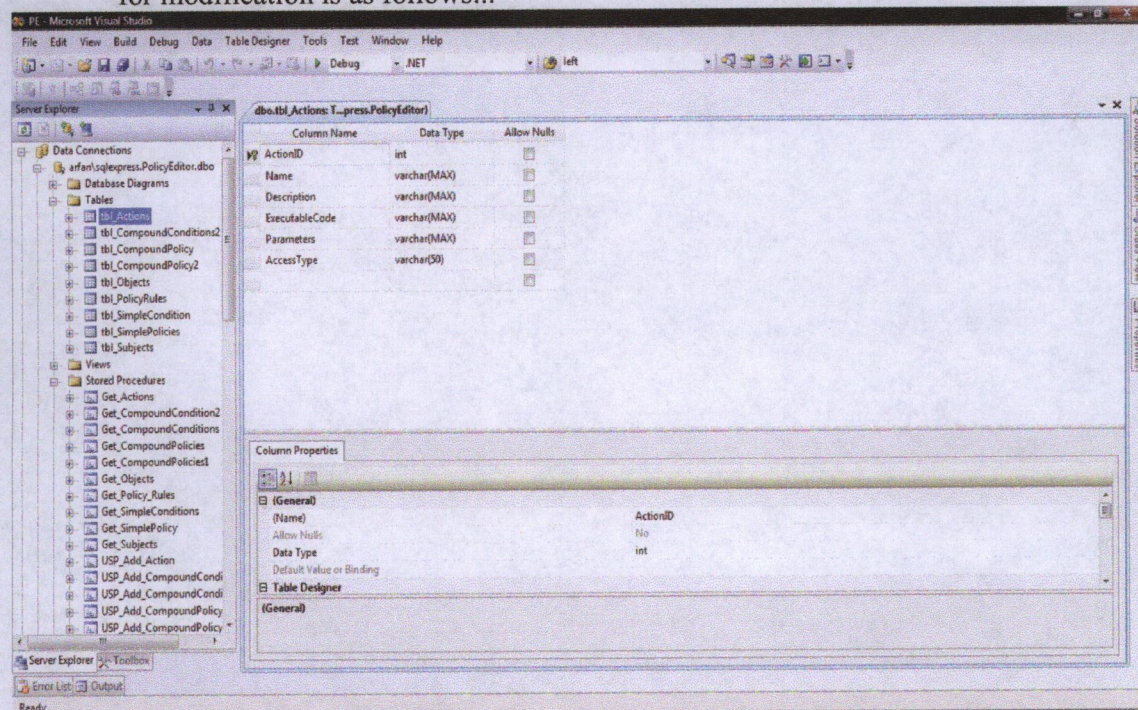
```
<?xml version="1.0"?>
<!--
Note: As an alternative to hand editing this file you can use the
web admin tool to configure settings for your application. Use
the Website->ASP.NET Configuration option in Visual Studio.
A full list of settings and comments can be found in
machine.config.comments usually located in
\Windows\Microsoft.Net\Framework\v2.0.50727\Config
-->
<configuration>
  <configSections>
    <sectionGroup name="system.web.extensions" type="System.Web.Configuration.SystemWebExtensionsSectionGroup, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
    <sectionGroup name="scripting" type="System.Web.Configuration.ScriptingSectionGroup, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
    <section name="scriptResourceHandler" type="System.Web.Configuration.ScriptingScriptResourceHandlerSection, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
    <sectionGroup name="webServices" type="System.Web.Configuration.ScriptingWebServicesSectionGroup, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
    <section name="jsonSerialization" type="System.Web.Configuration.ScriptingJsonSerializationSection, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
    <section name="profileService" type="System.Web.Configuration.ScriptingProfileServiceSection, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
    <section name="authenticationService" type="System.Web.Configuration.ScriptingAuthenticationServiceSection, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
    <section name="roleService" type="System.Web.Configuration.ScriptingRoleServiceSection, System.Web.Extensions, Version=3.5.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" />
  </sectionGroup>
</sectionGroup>
</sectionGroup>
</configSections>
<appSettings/>
<connectionStrings>
  <add name="ConnectionString" connectionString="Data Source=Abdulgade\SQLEXPRESS;Initial Catalog=PolicyEditor;User=sa;Password=123456" providerName="System.Data.SqlClient" />
</connectionStrings>
<system.web>
  <!--anonymousIdentification enabled="true"/-->
  <!--pages theme="Theme"/-->
</system.web>
</configuration>
```

Above is the configuration file and the highlighted line shows the database connection configuration which has been used throughout the application.

Database Tier:



As we discussed that SQL Server is integrated with .Net framework. It facilitates the developers by providing a proper interface where any modifications can be made to the databases. The screenshot above shows how to connect to the database before using it in Ms Visual Studio. Pane on the left side shows the database connection to be made or already made. Once use connected to the database and connection is successful then he/she can modify the database in the provided IDE. The main IDE for modification is as follows...



Above screenshot shows how user been provided with the IDE to perform any action on the database.

Client View of the System:

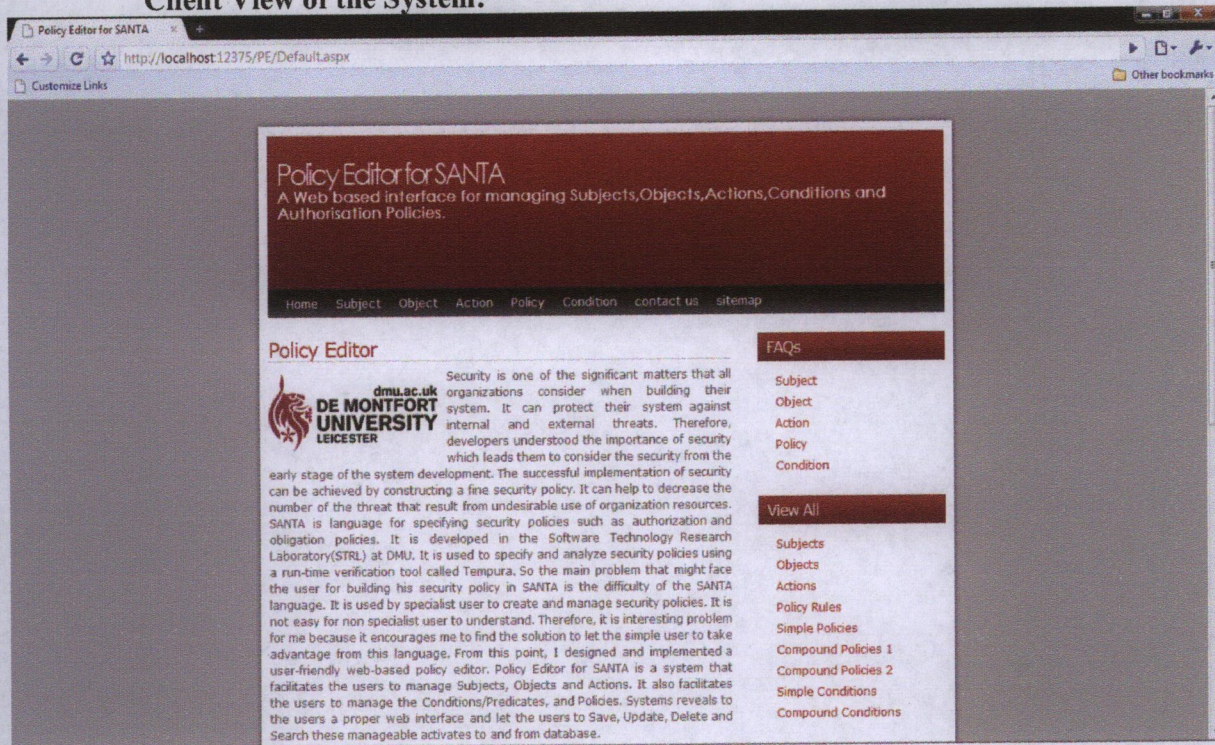


Figure 62: Client View of System Screen

6-3-System Requirements [Server Side]:

Policy Editor for SANTA will run on any windows platform above Windows 98. Following tools and technologies need to be installed before running the system.

- Service Pack 1 and Service Pack 2.
 - Internet Information Server (IIS).
 - Microsoft Visual Studio 2008 Express Edition or Microsoft Visual Web Developer 2008 Express Edition.
 - SQL Server 2005 as Database Server.
-
- Any Web Browser.

To run the system, just host it to the Web Server following the above mentioned tools and components. The Screenshots below shows how to run the Policy Editor.

6-4-Building/Compiling the System:

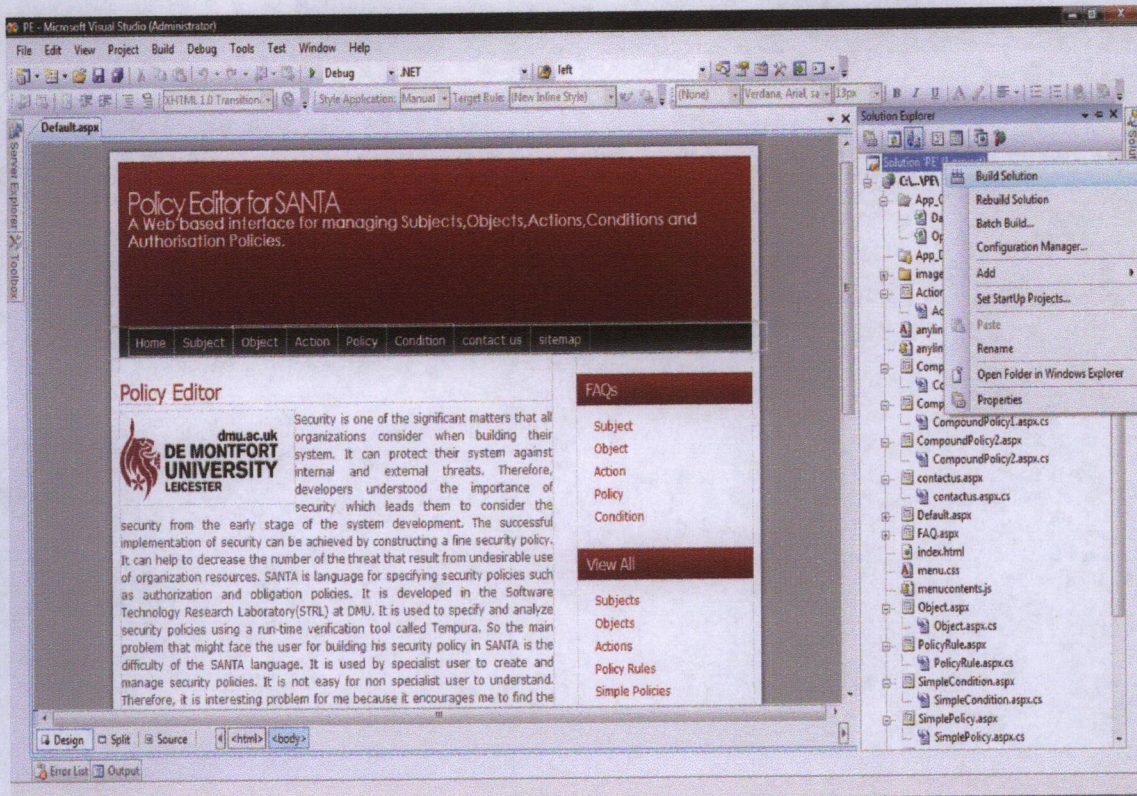


Figure 63: Compiling the System Screen

To build or compile the system simply right click the main project file and select the Build Website Option. This will build the website and confirm that the website is error and bug free and ready to run.

6-5-Running the System:

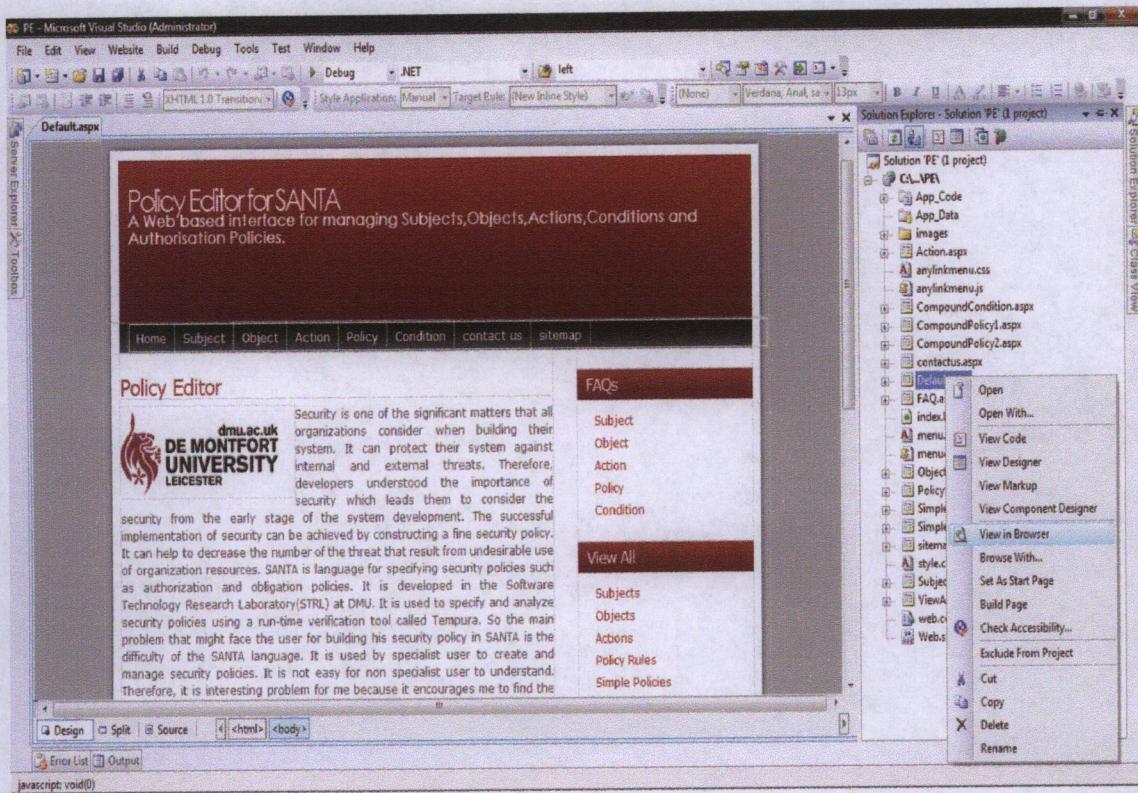


Figure 64: Running the System Screen

After Building or Compiling the website, next step is to run or execute it. To run or execute, right click on the main project file and select the View in Browser option. This will launch the website into the browser.

6-6-Policy Editor Screen Shots: Main Page:

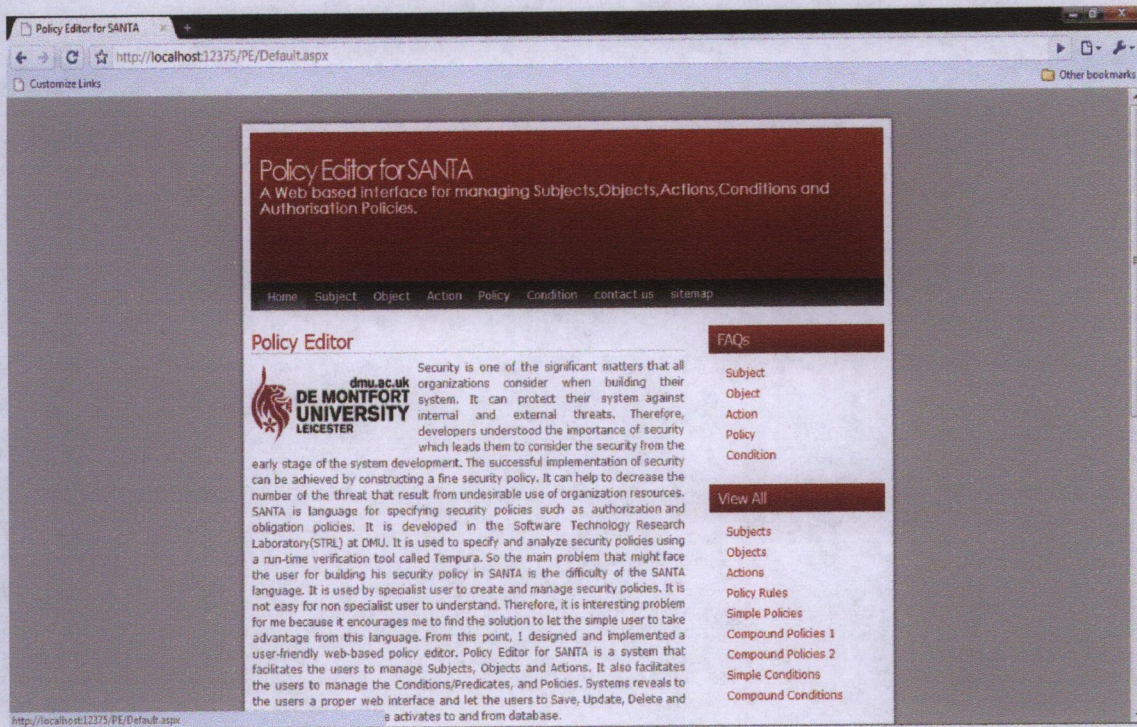


Figure 65: Main Page Screen

Here user can select any option to perform the activities. For example, after selecting the Subject menu user will be redirected to the subject page where he/she can add/delete/update and search the subjects. Same activities can be performed by selecting any of the menus at main page. Following screen shots will describe the pages and their functionalities.

Subject

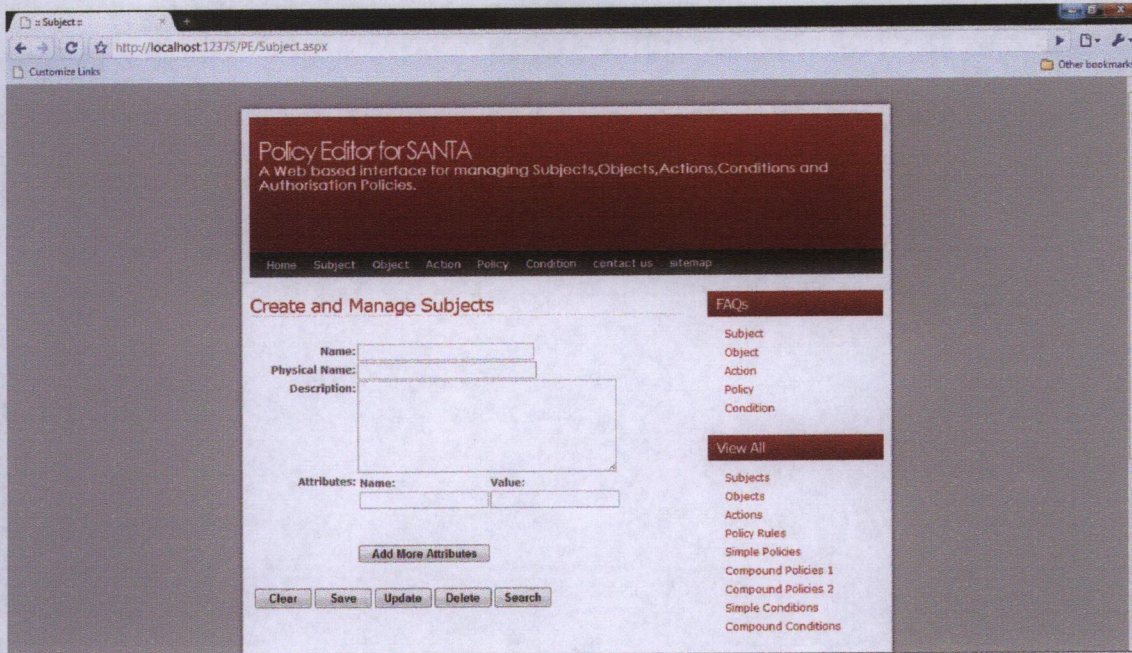


Figure 66: Subject Screen

All the fields would be provided to Add or update the Subject. If user wants to search the already saved subject he/she had to provide the subject name and system will return all the information based on that name. After searching user can update or delete the searched subject. User cannot add more than one subject with the same name.

Object:

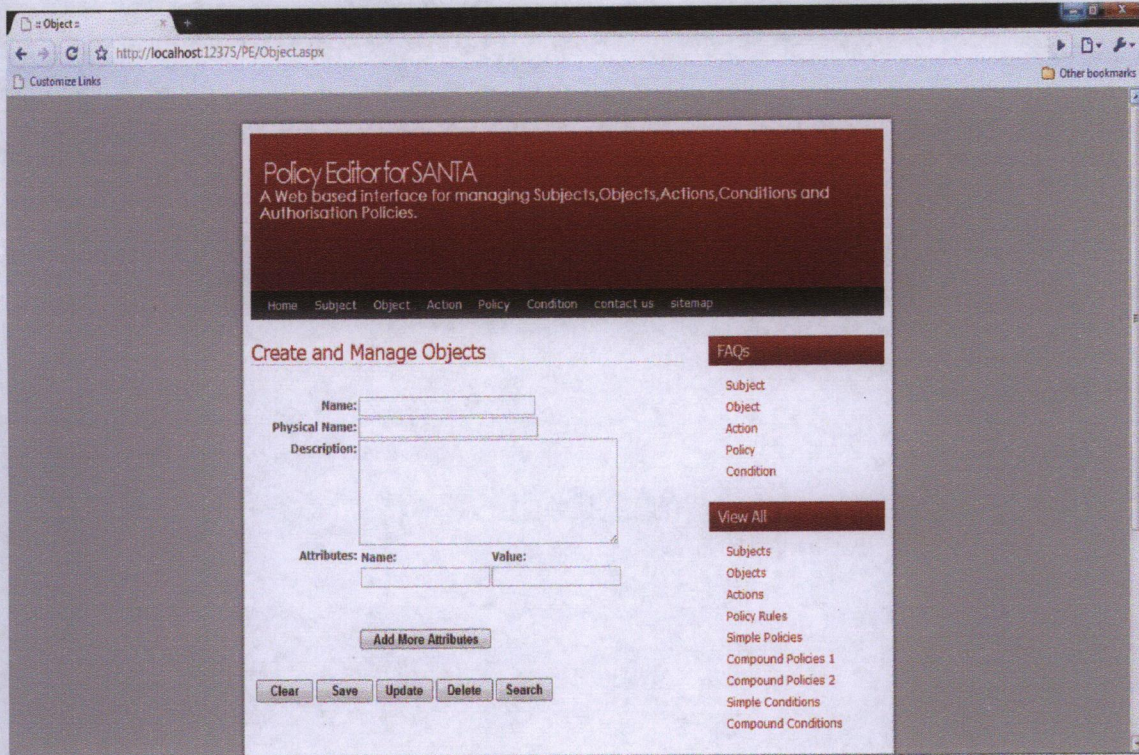


Figure 67: Object Screen

All the fields would be provided to Add or update the Object. If user wants to search the already saved subject he/she had to provide the object name and system will return all the information based on that name. After searching user can update or delete the searched object. User cannot add more than one object with the same name.

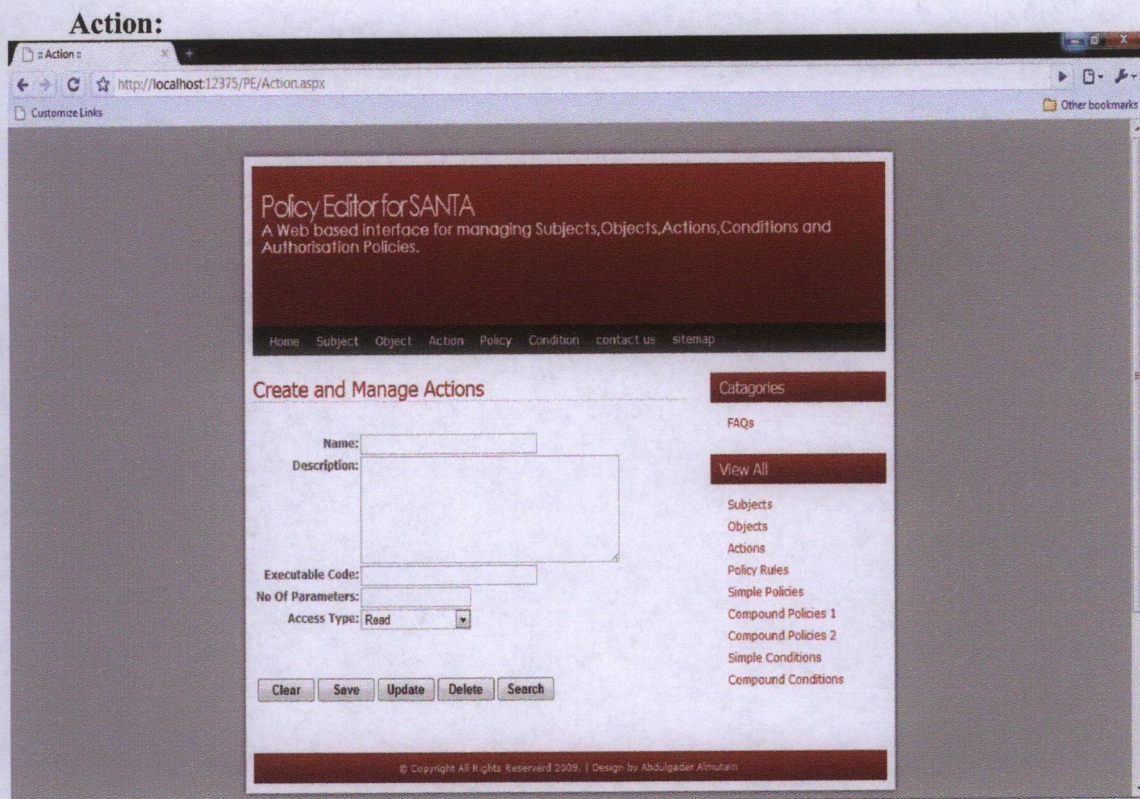


Figure 68: Action Screen

All the fields would be provided to Add or update an Action. If user wants to search the already saved subject he/she had to provide an action name and system will return all the information based on that name. After searching user can update or delete the searched action. User cannot add more than one action with the same name.

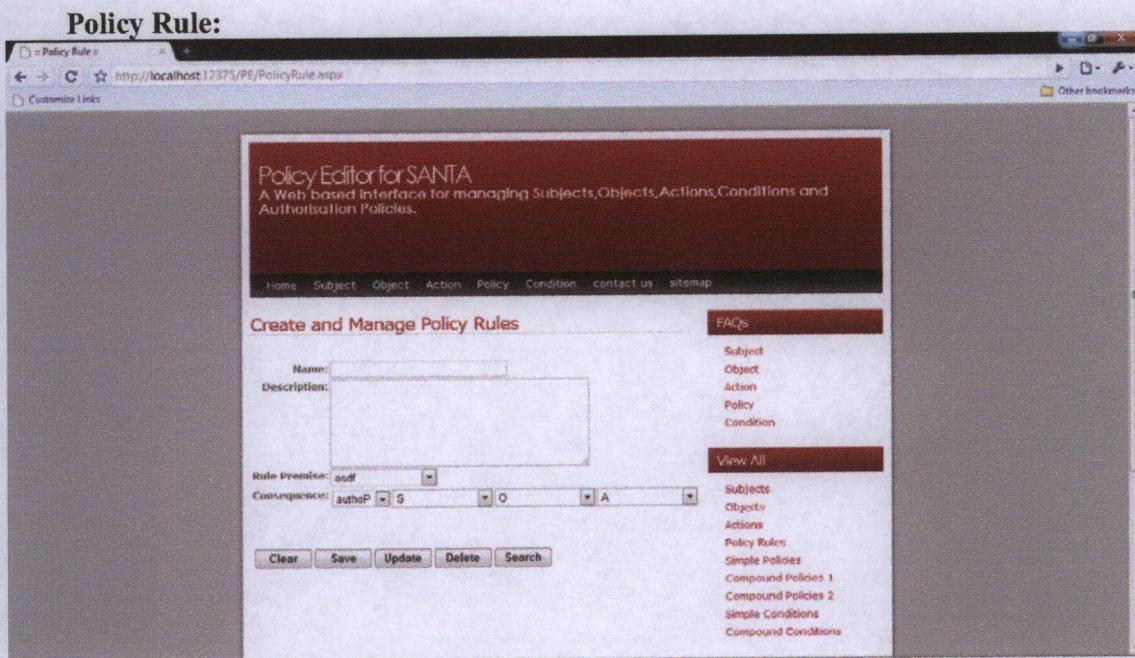


Figure 69: Policy Rule Screen

At first in the consequences field all the saved Subjects, Objects and Actions will be shown to the user. User can select anyone of his/her choice. All the fields would be provided to Add or update the Policy Rule. If user wants to search the already saved subject he/she had to provide the Policy Rule name and system will return all the information based on that name. After searching user can update or delete the searched Policy Rule. User cannot add more than one Policy Rules with the same name.

Simple Policy:

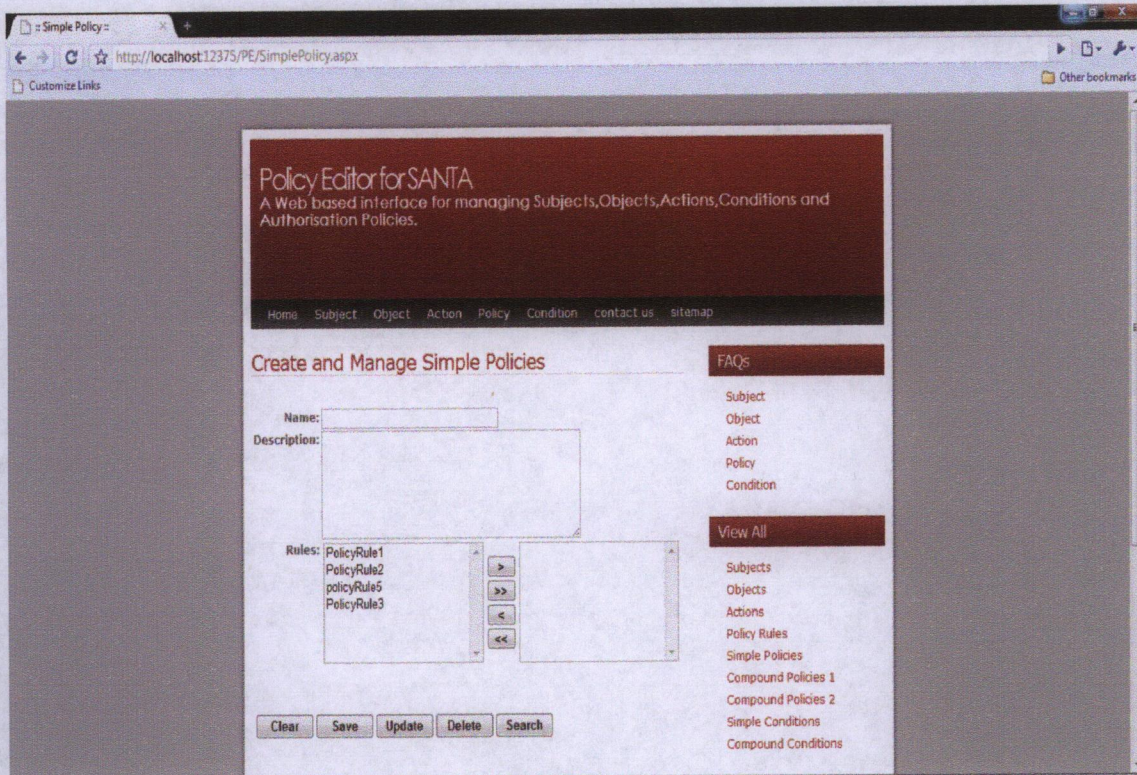


Figure 70:Simple Policy Screen

At first in the rules field all the saved policy rules will be shown to the user. User can select/deselect anyone or more than one of his/her choice. All the fields would be provided to Add or update the Simple Policy. If user wants to search the already saved Simple Policy he/she had to provide the Simple Policy name and system will return all the information based on that name. After searching user can update or delete the searched Simple Policy. User cannot add more than one Simple Policies with the same name.

Compound Policy1:

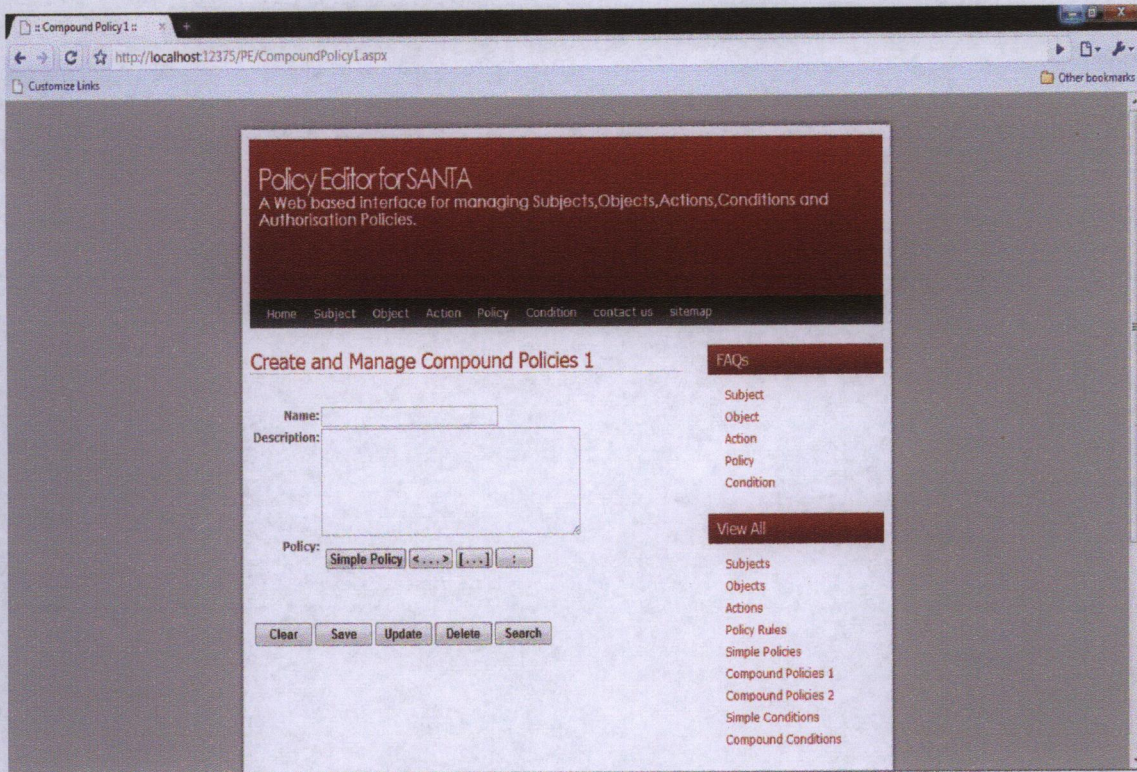


Figure 71: Compound Policy1 Screen

At first in the policy field all the fields would be hidden, User can press any of the buttons under the policy to add the compound policy1. Depending on the user selection, all the simple and compound policies will be shown to the user where user can select one of his own choices to define a new compound policy. All the fields would be provided to Add or update the Compound Policy1. If user wants to search the already saved Compound Policy1 he/she had to provide the Compound Policy1 name and system will return all the information based on that name. After searching user can update or delete the searched Compound Policy1. User cannot add more than one Compound Policy1 with the same name.

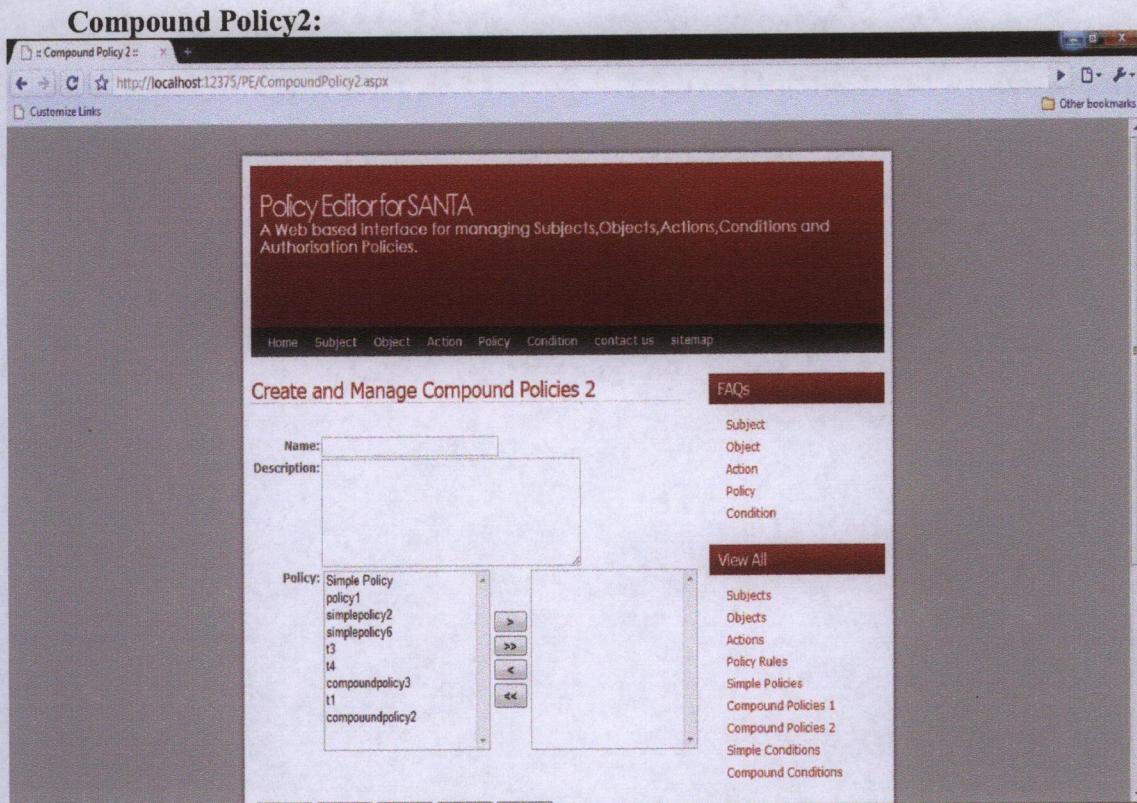


Figure 72: Compound Policy2 Screen

At first in the policy field all the saved compound and simple policies will be shown to the user. User can select/deselect anyone or more than one of his/her choice. All the fields would be provided to Add or update the Compound Policy2. If user wants to search the already saved Compound Policy2 he/she had to provide the Compound Policy2 name and system will return all the information based on that name. After searching user can update or delete the searched Compound Policy2. User cannot add more than one Compound Policy2 with the same name.

Simple Condition:

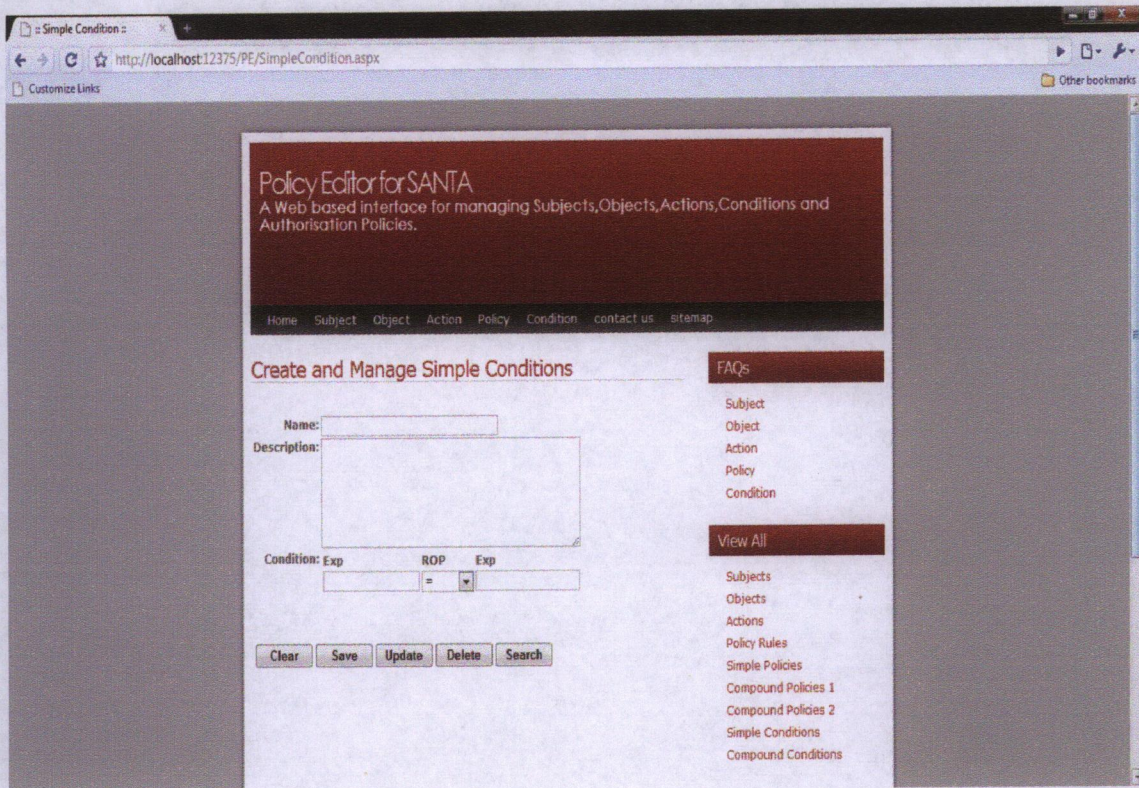


Figure 73: Simple Condition Screen

All the fields would be provided to Add or update a Simple Condition. If user wants to search the already saved Simple Condition he/she had to provide an action name and system will return all the information based on that name. After searching user can update or delete the searched Simple Condition. User cannot add more than one Simple Condition with the same name.

Compound Condition:

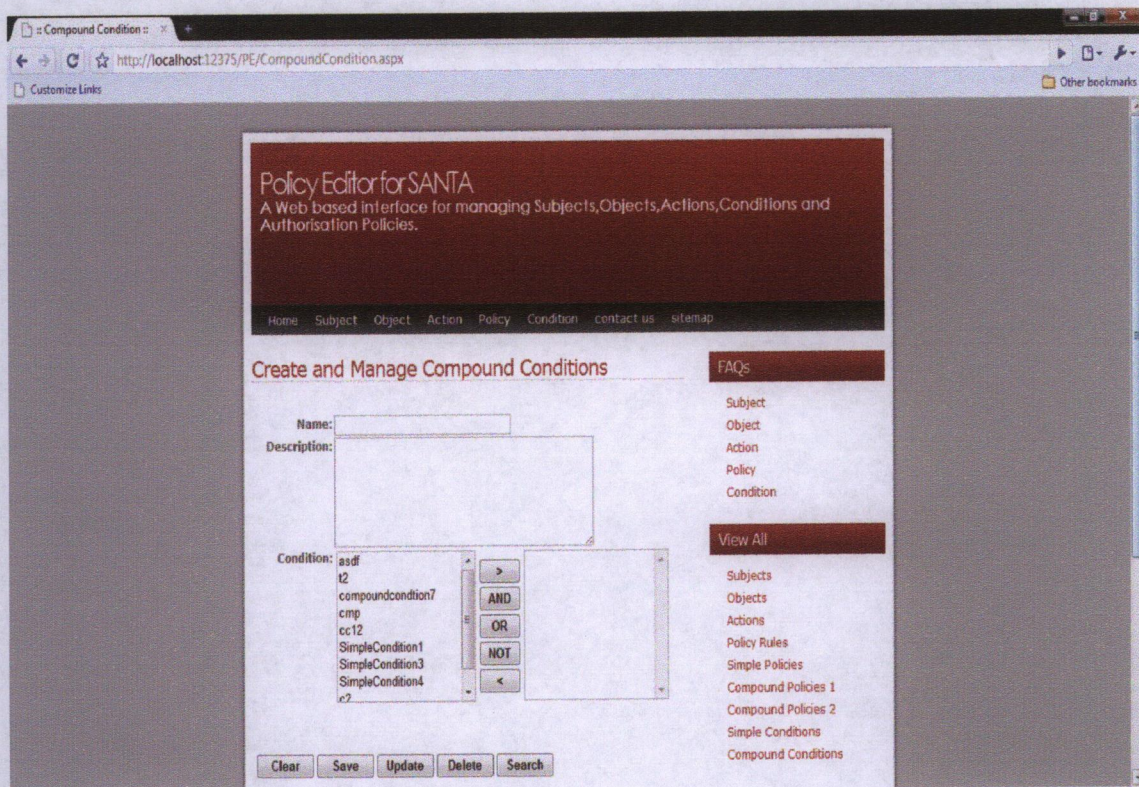


Figure 74: Compound Condition Screen

At first in the conditions field all the saved compound and simple conditions will be shown to the user. User can select/deselect anyone or more than one of his/her choice and user will be forced to put maximum one logical condition between any two selected conditions. All the fields would be provided to Add or update the Compound Condition. If user wants to search the already saved Compound Condition he/she had to provide the Compound Condition name and system will return all the information based on that name. After searching user can update or delete the searched Compound Condition. User cannot add more than one Compound Condition with the same name.

View All:

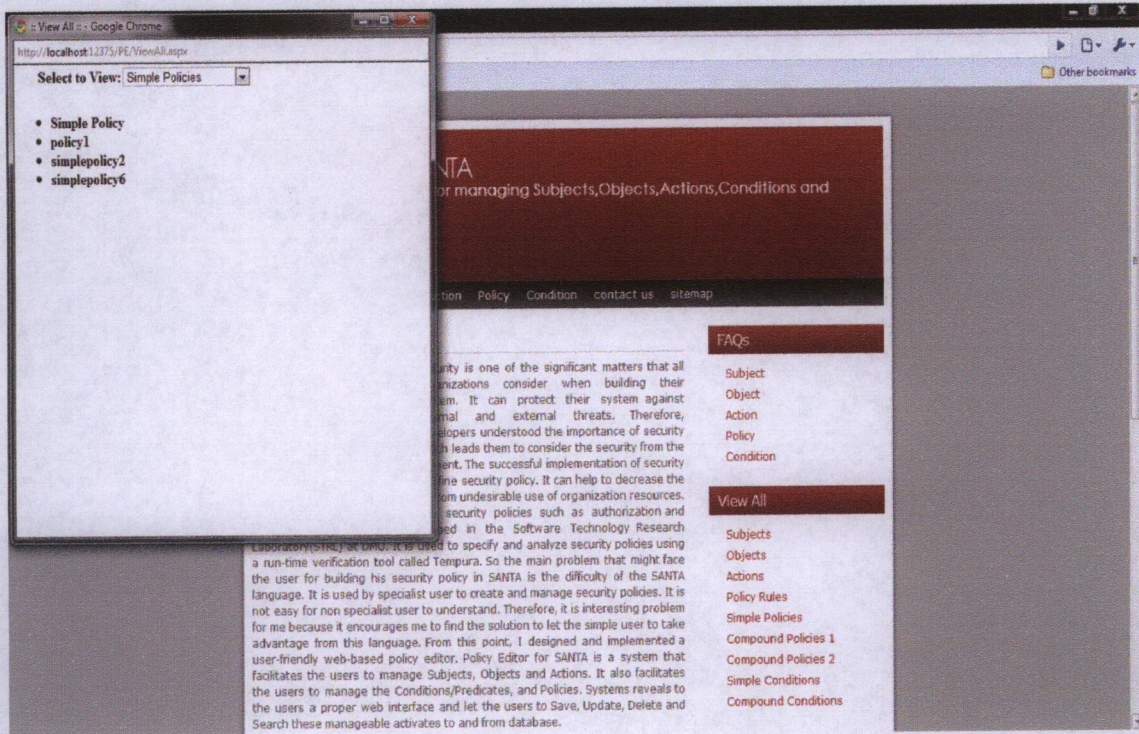


Figure 75: View All Screen

On this interface user has been facilitated to view the already existing Subjects, Objects, Actions, Policies and Conditions. After clicking any link under the heading of View All user will be prompted to a new window where he/she can select option to view anything which already saved to the database. This is useful to make the new users view anything they want to.

FAQs:

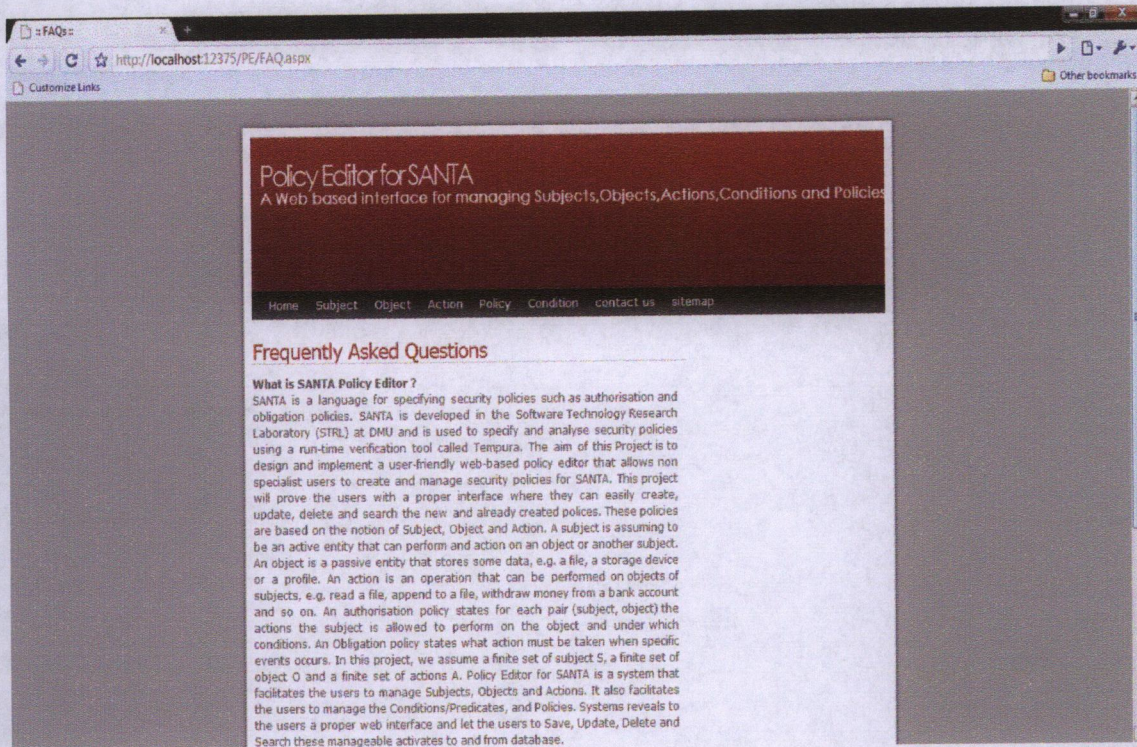


Figure 76: FAQs Screen

On this interface user can find any information which answers WHAT and HOW questions. On the FAQs page user has been given guidance about all the entities (Subject, Object, Actions, Policies, Conditions) of the system. User can select any entity from the links under heading of FAQs and then he/she would be redirected to the FAQs page.

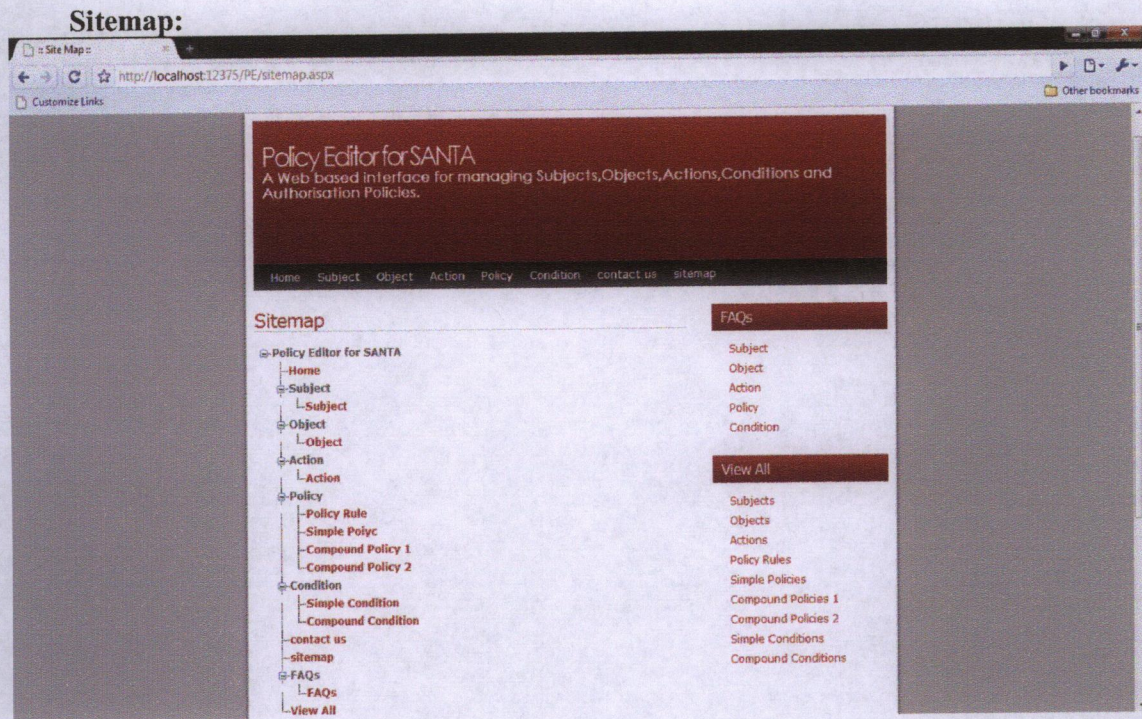


Figure 77: Site Map Screen

This interface shows the entire website sitemap.

Contact Us:

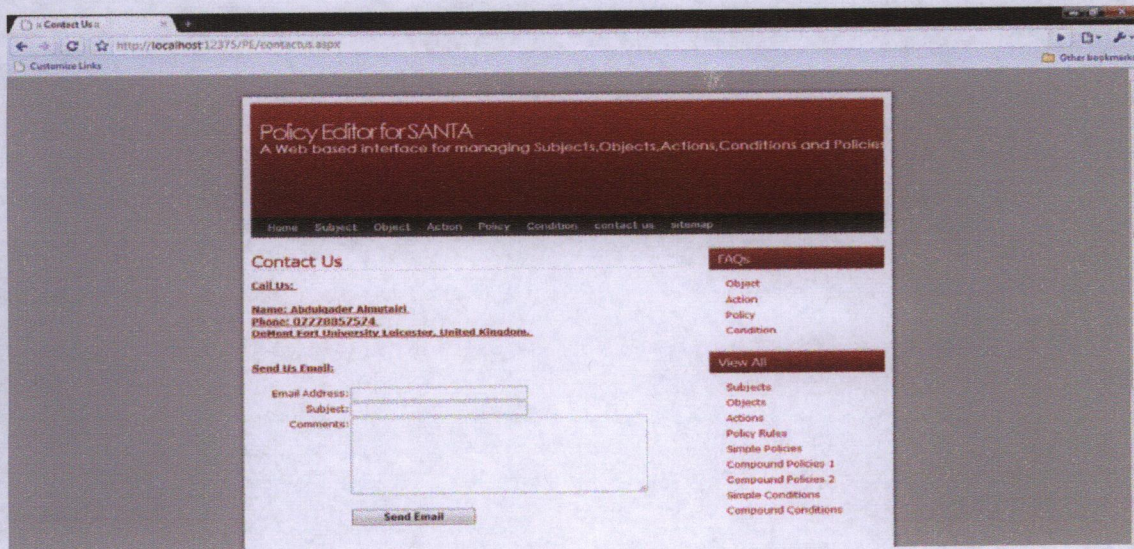


Figure 78: Contact US Screen

On this interface user can contact with the concerned authority through phone or email. User has been provided with a proper interface to contact via email. User will be responded on the provided email address which is a field in the form.

6-7-Summary:

In this chapter, the system was implemented based on the specification discussed in the earlier chapter. So the simple user can create and manage his policy easily. In addition, he/she can do some functionality on the policy created such as save, delete, update and search.

A web-based policy editor for SANTA	العنوان:
Al Mutairi, Abd Algder	المؤلف الرئيسي:
Siewe, Francois(Supervisor)	مؤلفين آخرين:
2009	التاريخ الميلادي:
Leicester	موقع:
1 - 105	الصفحات:
687206	رقم MD:
رسائل جامعية	نوع المحتوى:
English	اللغة:
رسالة ماجستير	الدرجة العلمية:
De Montfort University	الجامعة:
Faculty of Computing Sciences and Engineering	الكلية:
بريطانيا	الدولة:
Dissertations	قواعد المعلومات:
الأمن المعلوماتي، هندسة البرمجيات، الإنترنت، لغة البرامج	مواضيع:
https://search.mandumah.com/Record/687206	رابط:

Chapter 7: Testing and Evaluation

7-1- Overview:

This chapter describes the techniques used for the testing of Policy Editor for SANTA. Main focus is on Black Box and White Box Testing. As system is not very much complex in nature, these testing techniques are more suitable. These techniques enable to do the testing at both user and client end by considering all functional requirements.

7-2-Black Box Testing:

Black box testing is to test the system externally without going into details. This only requires the number of inputs to be used as test data and testers determine the output according to the specifications. The tests can be functional or non-functional, but usually black box testing is used for functional perspectives.

7-3- White Box Testing:

White box testing is to test the system both externally and internally considering all the details and states of the system at any given time. User does has to be focused on only inputs and outputs as test data but also test the flow of execution paths. For examples, the entire life time of an object and its attributes and functions. White box testing can target both functional and non-functional aspects of the system. This usually is been done at development team side.

7-4-Functional Testing

Sample Test Case:

Test Case	Description	Expected Result	Actual Result
T1_Add_Subject	Precondition: User is on the Subject Page and Provided All required Fields to Add the Subject. Enter: Press the Save Button.	Subject Added Successfully.	Subject Added Successfully.
T2_Add_Subject	Precondition: User is on the Subject Page and Provided All required Fields to Add the Subject and User Provided the already saved Subject Name. Enter: Press the Save Button	Subject with same name already exists.	Subject with same name already exists.

Table2: Test Case

7-5-Acceptance Testing:

Acceptance testing is done at the user end by considering the Project Supervisor as end user. This testing was successful and user was satisfied with the functionalities of the system.

7-6- Evaluation

Overall the system was evaluated as successful and provided all the desired functionalities. The functional, Acceptance testing concluded that the user was able to use the system in a very friendly manner and has got the desired results. A survey has been conducted by considering the participants as anonymous users. Following areas were the targets of survey.

1. Design of the System
2. Functionality according to Requirement Specifications
3. Ease of Understanding the System

7-6-1-Design of the System: [GUI]

The GUI (Graphical User Interface) and its cosmetics have been evaluated above average. Most of the male participants appreciated the design of the system by considering the professional GUI layouts. According to them the system's look and feel is professional and it makes feel the users that they are using a professional well designed and simple website. On the other hand, many female participants ranked the system design or GUI as average. They were very much concerned about the cosmetics of the system. The suggestions were given to add more contents and make it more colourful. At the end female participants were satisfied with the current GUI of the system which makes the overall evaluation of the system GUI above average. Moreover, complexity of system design is at very good level which means that any anonymous user with very little understand can use the system because of the fact that system GUI is simple and understandable and proper guide has been provided about how to use the system which can be found in FAQ section. This fact has been ranked very high by the participants.

Overall Evaluation Score: 70% Satisfied

7-6-2-Functionality according to Requirement Specifications:

The functional aspects of the system have been considered most important part. Participants have been given free hands to evaluate the functionality of the system. They were asked to be more keen and keep testing approach of to evaluate these functional aspects of the system. According to the participants, evaluation of the overall functionality of the system was regarded as excellent. We considered two to three

professionals for the evaluation of the functionality of system. Proper training was given about How and what concerns of system. Most of the participants were fully satisfied about the requirements communicated to them and the functionality of the system. According to them the current functionality of the system fully justifies the requirements. The flexibility and reusability of the system is much appreciated as it was communicated by considering the black box aspects of the system. Few suggestions were also given from the professional participants to improve the consistency, availability and durability of the system, which are in fact non-functional aspects of the system. But we will consider all the suggestions and comments to make the system robust and reliable. All these are considered as future works for system. Moreover, the level of complexity of the system, all the functional aspects can be easily understood and absorbed by anonymous users. Participants have also appreciated the technology used to develop the system which is state of art technology for both application development and database development. Overall the evaluation score of functional aspects of the system is excellent. Above 90 percent of the participants were satisfied with the functionality of the system.

Overall Evaluation Score: Above 90% Satisfied.

7-6-3-Ease of Understanding the System:

This section is very important from the end user point of view because we have considered the user of the system non-specialist and anonymous. The participants of this section were from different domains of the computer science. Participants were communicated about how they can use the system with maximum ease of use. After they use the system they found it very user friendly and easy to use. Most of the participants take no time to get familiar with system. The overall ranking for this section was excellent.

Overall Evaluation Score: Above 90% Satisfied

7-7-Summary

In this chapter, the system was tested by using the black and white box tools. In addition the system was evaluated by using the questionnaire method and the results are satisfied.

A web-based policy editor for SANTA	العنوان:
Al Mutairi, Abd Algder	المؤلف الرئيسي:
Siewe, Francois(Supervisor)	مؤلفين آخرين:
2009	التاريخ الميلادي:
Leicester	موقع:
1 - 105	الصفحات:
687206	رقم MD:
رسائل جامعية	نوع المحتوى:
English	اللغة:
رسالة ماجستير	الدرجة العلمية:
De Montfort University	الجامعة:
Faculty of Computing Sciences and Engineering	الكلية:
بريطانيا	الدولة:
Dissertations	قواعد المعلومات:
الأمن المعلوماتي، هندسة البرمجيات، الإنترنت، لغة البرامج	مواضيع:
https://search.mandumah.com/Record/687206	رابط:

Chapter 8: Conclusion and Future Work

8-1-Conclusion

In this project, we have focused on the importance of the security policy in the system. The successful implementation of security can be achieved by constructing a fine security policy. It can help to decrease the number of the threat that result from undesirable use of organization resources.

The overall aim of this Project was to design and implement a user-friendly web-based policy editor that allows non specialist users to create and manage security policies for SANTA.

SANTA is a language for specifying security policies such as authorization and obligation policies. This project proved the users with a proper interface where they can easily create, update, delete and search the new and already created polices. These policies are based on the notion of Subject, Object and Action. A subject is assuming to be an active entity that can perform and action on an object or another subject. An object is a passive entity that stores some data, e.g. a file, a storage device or a profile. An action is an operation that can be performed on objects of subjects, e.g. read a file, append to a file, withdraw money from a bank account and so on.

An authorization policy states for each pair (subject, object) the actions the subject is allowed to perform on the object and under which conditions. An Obligation policy states what action must be taken when specific events occurs. In this project, we assumed a finite set of subject S , a finite set of object O and a finite set of actions A . Policy Editor for SANTA is a system that facilitates the users to manage Subjects, Objects and Actions. It also facilitates the users to manage the Conditions/Predicates, and Policies. Systems reveals to the users a proper web interface and let the users to Save, Update, Delete and Search these manageable activates to and from database. It is very important for user to take advantage of SANTA language. However, it is very difficult language for simple user to use because it uses the ITL (interval temporal logic).

Therefore, this system enables users to create their policies without the need of understanding the ITL.

It was used finite state machine and JFLAB tool to model each part of the system. It is very beneficial to test the mobility of the system. In addition, the system was evaluated by using the questionnaire method to obtain the feedback from users. Hence the results were satisfactory as users were able to use the system easily and adapted very quickly to the system.

8-2- Future Work

The system can be enhanced further by allowing the user to generate a correct code in Tempura that corresponds to his policy. This code will be simulated using SPAT tool and in same time it can be executed by using a run-time verification tool called Tempura. In addition, the system can be improved by letting the user generate the XML code that corresponds to his policy.

Furthermore, in terms of improvement required in the design of the system as gathered from the questionnaire, there is couple of options to consider for future designs undertakings. These are summarized below:

- Make the GUI cosmetics more prominent by considering all aspects of look and feel of website. This would be done by using a dedicated designing tool.
- Making assure the non-functional requirements of the system. Proper considerations for making the system reliable, available and durable.

A web-based policy editor for SANTA	العنوان:
Al Mutairi, Abd Algder	المؤلف الرئيسي:
Siewe, Francois(Supervisor)	مؤلفين آخرين:
2009	التاريخ الميلادي:
Leicester	موقع:
1 - 105	الصفحات:
687206	رقم MD:
رسائل جامعية	نوع المحتوى:
English	اللغة:
رسالة ماجستير	الدرجة العلمية:
De Montfort University	الجامعة:
Faculty of Computing Sciences and Engineering	الكلية:
بريطانيا	الدولة:
Dissertations	قواعد المعلومات:
الأمن المعلوماتي، هندسة البرمجيات، الإنترنت، لغة البرامج	مواضيع:
https://search.mandumah.com/Record/687206	رابط:

Abstract:

Security is one of the significant matters that all organizations consider when building their system. It can protect their system against internal and external threats. Therefore, developers understood the importance of security which leads them to consider the security from the early stage of the system development.

The successful implementation of security can be achieved by constructing an efficient security policy. This can help to decrease the number of the threat that result from undesirable use of organization resources.

SANTA is language for specifying security policies such as authorization and obligation policies. It is developed in the Software Technology Research Laboratory (STRL) at DMU and was used to specify and analyze security policies using a run-time verification tool called Tempura. However, the main problem that might face the non specialist users is that they must have enough knowledge of ITL (Interval Temporal Logic) to use SANTA language because the SANTA is based on ITL. Professional users have used the SANTA to create and manage security policies. There exists a policy editor for SANTA developed in Java. The use of this editor requires the users to be familiar with the concrete syntax of SANTA. Given that SANTA language has a strong link with the ITL, a substantial knowledge of ITL is also required. These make it difficult for non specialist users, with limited or no knowledge in temporal logic, to use SANTA.

To make SANTA accessible also to non specialist users, I have designed and implemented a user-friendly, graphical and web-based policy editor for SANTA. The editor is implemented using a combination of web technologies: HTML, ASP.net and SQL server 2005. The main functionalities of the editor include:

- 1- Management of subjects, objects and actions.
- 2- Management of conditions.
- 3- Management of policy rules.
- 4- Management of simple policies.
- 5- Management of compound policies.

A web-based policy editor for SANTA	العنوان:
Al Mutairi, Abd Algder	المؤلف الرئيسي:
Siewe, Francois(Supervisor)	مؤلفين آخرين:
2009	التاريخ الميلادي:
Leicester	موقع:
1 - 105	الصفحات:
687206	رقم MD:
رسائل جامعية	نوع المحتوى:
English	اللغة:
رسالة ماجستير	الدرجة العلمية:
De Montfort University	الجامعة:
Faculty of Computing Sciences and Engineering	الكلية:
بريطانيا	الدولة:
Dissertations	قواعد المعلومات:
الأمن المعلوماتي، هندسة البرمجيات، الإنترنت، لغة البرامج	مواضيع:
https://search.mandumah.com/Record/687206	رابط:

Table of Content

Acknowledgements.....	1
Abstract.....	2
Table of Content.....	3
Chapter 1: Introduction.....	10
1-1 Motivation.....	10
1-2 Original Contribution.....	11
1-3 Objective.....	11
1-4 Methodology to reach the Project.....	11
1-5 Project Planning.....	12
1-6 Resource of the Project.....	12
1-7 Tools and Technologies Used.....	13
1-8 Dissertation Outline.....	13
Chapter 2: Literature Review.....	14
2-1 Overview.....	14
2-2 Security Requirements.....	14
2-3 The Fundamental Components.....	14
2-3-1 Confidentiality.....	14
2-3-2 Integrity.....	15
2-3-2-1 Prevention Mechanisms.....	15
2-3-2-2 Detection Mechanisms.....	15

2-3-3 Availability.....	16
2-4 Security Policy.....	17
2-4-1 Definition.....	17
2-4-2 Security Model.....	17
2-4-2-1 Overview.....	17
2-4-2-2 Bell-LaPadula Policy Model.....	17
2-4-2-3 Clark-Wilson Model.....	18
2-4-2-4 Chinese Wall Model.....	18
2-5 Security Policy Language.....	19
2-5-1 Overview.....	19
2-5-2 High-Level Policy Languages.....	19
2-5-3 Low-Level Policy Languages.....	19
2-6 Policy Management.....	19
2-7 Repudiation.....	20
2-8 Development of Security Policy.....	20
2-9 Security Policy Violation.....	20
2-9-1 Interception.....	21
2-9-2 Modification/Iteration.....	21
2-9-3 Masquerading/Spoofing.....	21
2-9-4 Interruption.....	21
2-9-5 Fabrications.....	22

Chapter 3: SANTA Language	22
3-1 Overview.....	22
3-2 Definition.....	22
3-3 Policies in SANTA.....	22
3-4- Syntax of SANTA Policy.....	22
3-4-1- Simple Condition.....	22
3-4-2-Compound Condition.....	23
3-4-3-Policy Rule	23
3-4-3-1 Authorization.....	23
3-4-4 Simple Policy.....	25
3-4-5- Compound Policy.....	25
3-4-6-Delegation.....	25
3-4-7- Obligation.....	26
3-4-8- Types of Policies in SANTA.....	26
3-6- Policy Specification.....	26
3-7-Policy Composition.....	27
3-9 Summary.....	27
Chapter 4: Modelling of The System By (FSM)	28
4-1 Introduction.....	28
4-2 Definition.....	28
4-3 JFLAP.....	29

4-4 Design Model of States.....	29
4-5 Testing the State Machine.....	29
4-6 Summary.....	39
Chapter 5: Requirements, Analysis and Design.....	40
5-1 Overview.....	40
5-2 Problem Domain Requirements.....	40
5-3 System Requirements.....	40
5-4 UML Diagrams.....	41
5-4-1 Use Case Diagram.....	41
5-4-2 Class Diagram.....	49
5-4-3 Sequence Diagram.....	50
5-5 Flow Chart of the System.....	69
5-6 Summary.....	69
Chapter 6: Implementation.....	70
6-1 Overview.....	70
6-2 System Requirements (Client Side).....	70
6-3 System Requirements (Server Side).....	70
6-4 Building/Compiling the System.....	71
6-5 Running the System.....	72
6-6 The System Screens Shots.....	73
6-7 Summary.....	86

Chapter 7: Testing and Evaluation	87
7-1 Overview.....	87
7-2 Black Box Testing.....	87
7-3 White Box Testing.....	87
7-4 Functional Testing.....	88
7-5 Acceptance Testing.....	88
7-6 Evaluation.....	89
7-6-1 Design of the System (GUI).....	89
7-6-2 Functionality According to Requirements Specification.....	89
7-6-3 Ease of Understanding of the System.....	90
7-7 Summary.....	90
Chapter 8: Conclusion and Future Work	91
8-1 Conclusion.....	91
8-2 Future Work.....	92
References	93

List of Figure and Table

Figure 1: Information Security Aspects.....	16
Figure 2: Basic Model of (FSM).....	29
Figure 3: Simple Condition Model (FSM).....	30
Figure 4: Compound Condition Model (FSM).....	31
Figure 5: Policy Rule Model (FSM).....	32
Figure 6: Simple Policy Model (FSM).....	33
Figure 7: Compound Policy1 Model (FSM).....	34
Figure 8: Compound Policy2 Model (FSM).....	35
Figure 9: Subject and Object Model (FSM).....	36
Figure 10: Action Model (FSM).....	36
Figure 11: Simple Condition Model Test.....	37
Figure 12: Compound Condition Model Test.....	38
Figure 13: Action Model Test.....	39
Figure 14: System Level.....	41
Figure 15: Action Use Case Diagram.....	42
Figure 16: Subject Use Case Diagram.....	42
Figure 17: Object Use Case Diagram.....	43
Figure 18: Policy Rule Use Case Diagram.....	44
Figure 19: Simple Policy Use Case Diagram.....	45
Figure 20: Compound Policy1 Use Case Diagram.....	46
Figure 21: Compound Policy2 Use Case Diagram.....	47
Figure 22: Simple Condition Use Case Diagram.....	48
Figure 23: Compound Condition Use Case Diagram.....	49
Figure 24: System Class Diagram.....	50
Figure 25: Functionality of Action Diagram.....	51
Figure 26: Functionality of Subject Diagram.....	53
Figure 27: Functionality of Object Diagram.....	55
Figure 28: Functionality of Policy Rule Diagram.....	57
Figure 29: Functionality of Simple Policy Diagram.....	59
Figure 30: Functionality of Compound Policy1 Diagram.....	61
Figure 31: Functionality of Compound policy 2Diagram.....	63
Figure 32: Functionality of Simple Condition Diagram.....	65
Figure 33: Functionality of Compound Condition Diagram.....	67
Figure 34: Flow Chart for the System.....	69
Figure 35: Client View of System Screen.....	70
Figure 36: Compiling the System Screen.....	71
Figure 37: Running the System Screen.....	72
Figure 38: Main Page Screen.....	73
Figure 39: Subject Screen.....	74
Figure 40: Object Screen.....	75
Figure 41: Action Screen.....	76
Figure 42: Policy Rule Screen.....	77
Figure 43: Simple Policy Screen.....	78
Figure 44: Compound Policy1 Screen.....	79
Figure 45: Compound policy2 Screen.....	80
Figure 46: Simple Condition Screen.....	81
Figure 47: Compound Condition Screen.....	82
Figure 48: View All Screen.....	83
Figure 49: FAQs Screen.....	84
Figure 50: Site Map Screen.....	85
Figure 51: Contact Us Screen.....	85
Table 1: Project Plan.....	12
Table 2: Test Case.....	88

A web-based policy editor for SANTA	العنوان:
Al Mutairi, Abd Algder	المؤلف الرئيسي:
Siewe, Francois(Supervisor)	مؤلفين آخرين:
2009	التاريخ الميلادي:
Leicester	موقع:
1 - 105	الصفحات:
687206	رقم MD:
رسائل جامعية	نوع المحتوى:
English	اللغة:
رسالة ماجستير	الدرجة العلمية:
De Montfort University	الجامعة:
Faculty of Computing Sciences and Engineering	الكلية:
بريطانيا	الدولة:
Dissertations	قواعد المعلومات:
الأمن المعلوماتي، هندسة البرمجيات، الإنترنت، لغة البرامج	مواضيع:
https://search.mandumah.com/Record/687206	رابط:

A Web-based Policy Editor for SANTA

MSC SOFTWARE ENGINEERING
COMP5314

Dissertation

SUPERVISED BY Francois Siewe
Done by Abdugder Almutairi

P06005561

2009
DeMontfort
university

Acknowledgements

Firstly, I am grateful to Allah for giving me the strength and courage and for enabling me complete this dissertation.

I thank my project supervisor Dr Francois Siewe for all his support and guidance during the course of writing this dissertation.

I also thank Prof. Zedan for his help during my dissertation.

I also thank the second supervisor Dr Helge T. Janicke for his help during my dissertation.

I thank all staff in DMU and STRL who helps me to finish my dissertation.

I thank my entire family for their support, encouragement and during my MSc course especially my parents.

Finally, I would like to special thank my wife and my children for being patient while I was doing my project, without their patience most of these work wouldn't have been accomplished.

Abstract:

Security is one of the significant matters that all organizations consider when building their system. It can protect their system against internal and external threats. Therefore, developers understood the importance of security which leads them to consider the security from the early stage of the system development.

The successful implementation of security can be achieved by constructing an efficient security policy. This can help to decrease the number of the threat that result from undesirable use of organization resources.

SANTA is language for specifying security policies such as authorization and obligation policies. It is developed in the Software Technology Research Laboratory (STRL) at DMU and was used to specify and analyze security policies using a run-time verification tool called Tempura. However, the main problem that might face the non specialist users is that they must have enough knowledge of ITL (Interval Temporal Logic) to use SANTA language because the SANTA is based on ITL. Professional users have used the SANTA to create and manage security policies. There exists a policy editor for SANTA developed in Java. The use of this editor requires the users to be familiar with the concrete syntax of SANTA. Given that SANTA language has a strong link with the ITL, a substantial knowledge of ITL is also required. These make it difficult for non specialist users, with limited or no knowledge in temporal logic, to use SANTA.

To make SANTA accessible also to non specialist users, I have designed and implemented a user-friendly, graphical and web-based policy editor for SANTA. The editor is implemented using a combination of web technologies: HTML, ASP.net and SQL server 2005. The main functionalities of the editor include:

- 1- Management of subjects, objects and actions.
- 2- Management of conditions.
- 3- Management of policy rules.
- 4- Management of simple policies.
- 5- Management of compound policies.

Table of Content

Acknowledgements.....	1
Abstract.....	2
Table of Content.....	3
Chapter 1: Introduction.....	10
1-1 Motivation.....	10
1-2 Original Contribution.....	11
1-3 Objective.....	11
1-4 Methodology to reach the Project.....	11
1-5 Project Planning.....	12
1-6 Resource of the Project.....	12
1-7 Tools and Technologies Used.....	13
1-8 Dissertation Outline.....	13
Chapter 2: Literature Review.....	14
2-1 Overview.....	14
2-2 Security Requirements.....	14
2-3 The Fundamental Components.....	14
2-3-1 Confidentiality.....	14
2-3-2 Integrity.....	15
2-3-2-1 Prevention Mechanisms.....	15
2-3-2-2 Detection Mechanisms.....	15

2-3-3 Availability.....	16
2-4 Security Policy.....	17
2-4-1 Definition.....	17
2-4-2 Security Model.....	17
2-4-2-1 Overview.....	17
2-4-2-2 Bell-LaPadula Policy Model.....	17
2-4-2-3 Clark-Wilson Model.....	18
2-4-2-4 Chinese Wall Model.....	18
2-5 Security Policy Language.....	19
2-5-1 Overview.....	19
2-5-2 High-Level Policy Languages.....	19
2-5-3 Low-Level Policy Languages.....	19
2-6 Policy Management.....	19
2-7 Repudiation.....	20
2-8 Development of Security Policy.....	20
2-9 Security Policy Violation.....	20
2-9-1 Interception.....	21
2-9-2 Modification/Iteration.....	21
2-9-3 Masquerading/Spoofing.....	21
2-9-4 Interruption.....	21
2-9-5 Fabrications.....	22

Chapter 3: SANTA Language	22
3-1 Overview.....	22
3-2 Definition.....	22
3-3 Policies in SANTA.....	22
3-4- Syntax of SANTA Policy.....	22
3-4-1- Simple Condition.....	22
3-4-2-Compound Condition.....	23
3-4-3-Policy Rule	23
3-4-3-1 Authorization.....	23
3-4-4 Simple Policy.....	25
3-4-5- Compound Policy.....	25
3-4-6-Delegation.....	25
3-4-7- Obligation.....	26
3-4-8- Types of Policies in SANTA.....	26
3-6- Policy Specification.....	26
3-7-Policy Composition.....	27
3-9 Summary.....	27
Chapter 4: Modelling of The System By (FSM)	28
4-1 Introduction.....	28
4-2 Definition.....	28
4-3 JFLAP.....	29

4-4 Design Model of States.....	29
4-5 Testing the State Machine.....	29
4-6 Summary.....	39
Chapter 5: Requirements, Analysis and Design.....	40
5-1 Overview.....	40
5-2 Problem Domain Requirements.....	40
5-3 System Requirements.....	40
5-4 UML Diagrams.....	41
5-4-1 Use Case Diagram.....	41
5-4-2 Class Diagram.....	49
5-4-3 Sequence Diagram.....	50
5-5 Flow Chart of the System.....	69
5-6 Summary.....	69
Chapter 6: Implementation.....	70
6-1 Overview.....	70
6-2 System Requirements (Client Side).....	70
6-3 System Requirements (Server Side).....	70
6-4 Building/Compiling the System.....	71
6-5 Running the System.....	72
6-6 The System Screens Shots.....	73
6-7 Summary.....	86

Chapter 7: Testing and Evaluation	87
7-1 Overview.....	87
7-2 Black Box Testing.....	87
7-3 White Box Testing.....	87
7-4 Functional Testing.....	88
7-5 Acceptance Testing.....	88
7-6 Evaluation.....	89
7-6-1 Design of the System (GUI).....	89
7-6-2 Functionality According to Requirements Specification.....	89
7-6-3 Ease of Understanding of the System.....	90
7-7 Summary.....	90
Chapter 8: Conclusion and Future Work	91
8-1 Conclusion.....	91
8-2 Future Work.....	92
References	93

List of Figure and Table

Figure 1: Information Security Aspects.....	16
Figure 2: Basic Model of (FSM).....	29
Figure 3: Simple Condition Model (FSM).....	30
Figure 4: Compound Condition Model (FSM).....	31
Figure 5: Policy Rule Model (FSM).....	32
Figure 6: Simple Policy Model (FSM).....	33
Figure 7: Compound Policy1 Model (FSM).....	34
Figure 8: Compound Policy2 Model (FSM).....	35
Figure 9: Subject and Object Model (FSM).....	36
Figure 10: Action Model (FSM).....	36
Figure 11: Simple Condition Model Test.....	37
Figure 12: Compound Condition Model Test.....	38
Figure 13: Action Model Test.....	39
Figure 14: System Level.....	41
Figure 15: Action Use Case Diagram.....	42
Figure 16: Subject Use Case Diagram.....	42
Figure 17: Object Use Case Diagram.....	43
Figure 18: Policy Rule Use Case Diagram.....	44
Figure 19: Simple Policy Use Case Diagram.....	45
Figure 20: Compound Policy1 Use Case Diagram.....	46
Figure 21: Compound Policy2 Use Case Diagram.....	47
Figure 22: Simple Condition Use Case Diagram.....	48
Figure 23: Compound Condition Use Case Diagram.....	49
Figure 24: System Class Diagram.....	50
Figure 25: Functionality of Action Diagram.....	51
Figure 26: Functionality of Subject Diagram.....	53
Figure 27: Functionality of Object Diagram.....	55
Figure 28: Functionality of Policy Rule Diagram.....	57
Figure 29: Functionality of Simple Policy Diagram.....	59
Figure 30: Functionality of Compound Policy1 Diagram.....	61
Figure 31: Functionality of Compound policy 2Diagram.....	63
Figure 32: Functionality of Simple Condition Diagram.....	65
Figure 33: Functionality of Compound Condition Diagram.....	67
Figure 34: Flow Chart for the System.....	69
Figure 35: Client View of System Screen.....	70
Figure 36: Compiling the System Screen.....	71
Figure 37: Running the System Screen.....	72
Figure 38: Main Page Screen.....	73
Figure 39: Subject Screen.....	74
Figure 40: Object Screen.....	75
Figure 41: Action Screen.....	76
Figure 42: Policy Rule Screen.....	77
Figure 43: Simple Policy Screen.....	78
Figure 44: Compound Policy1 Screen.....	79
Figure 45: Compound policy2 Screen.....	80
Figure 46: Simple Condition Screen.....	81
Figure 47: Compound Condition Screen.....	82
Figure 48: View All Screen.....	83
Figure 49: FAQs Screen.....	84
Figure 50: Site Map Screen.....	85
Figure 51: Contact Us Screen.....	85
Table 1: Project Plan.....	12
Table 2: Test Case.....	88

Chapter 1: Introduction

1-1- Motivation

The expansion of the internet and networks in the world led to increase the number of security threats and breaches. Therefore, security should be constructed in the early stage for system development [11]. Hence, the developers are focusing on security requirements and how to achieve three key aspects or requirements: confidentiality, integrity and availability.

The major purpose of the security requirements is the prevention of the data resources from threats. This is achieved by setting up constraints that define individuals permitted access to the data assets of a system. However, confidentiality requires that only authorized individuals are allowed to access computer system resources, whereas integrity describes the prevention of an unofficial modification of data. Availability of required resource or information is its ability to be used [1].

Security policy is applied as a basis of configuration and auditing of computer systems and networks, while upholding obedience with the policy requirements. These allow later implementation of developmental, operational guidelines and introduction of user access monitoring regulations [6].

Security is a major issue for organisation which deals directly with internet by doing online business or providing services. The leak of security can facilitate the increase exposure of the threat to the internet which can cause loss or damage of the information or equipment or threat human life in some critical system.

This project encourages me to solve the problem that result from the difficulty of using SANTA language for non specialist user. It will deliver a friendly editor that can allow the user to create his policy easily. This policy will be created by using SANTA structure.

1-2- Original Contribution:

In this dissertation, I have designed and implemented a user-friendly web-based policy editor that allows non specialist users (normal users) to create and manage security policies in SANTA. It is a language for specifying security policies such as authorization and obligation policies. It is developed in the Software Technology Research Laboratory (STRL) at DMU. Policies specified in SANTA can be analyzed using a run-time verification tool called Tempura.

This editor has different interfaces that allow the user to create simple and compound conditions, policy rules, simple policies and compound policies and. In addition, it has interfaces for managing subjects, objects and actions. Furthermore, it also allows the user to conduct functionality upgrades such as modifying, deleting the existing policies and adding new policies. All interfaces connected with database that can manage all policies.

1-3- Objectives

There are many objectives for my project:

- It helps to become skilled at programming languages.
- It helps to understand the concept of the security requirements and security policy.
- It helps to identify the security policy language called SANTA.
- It helps to apply some of my knowledge that learned from the modules.

1-4- Methodology to reach the project

In the analysis stage, the UML (Unified Modeling Language) were used to build the structure of the software. So it helps to identify the functional and non functional requirement. In addition, the modeling of my project was constructed by using JFLAB tools. The ASP.net software with SQL server 2005 was used to implement the project because I found it to be suitable software that has several features. In the later stage Black and white test method were used.

1-5- Project Planning Table

It is important for me to organize my work from the early stage until the final stage. So it helps me to finish the project on time.

ID	Task Name	Start	Finish	Duration
1	Choose Topic and approval by supervisors	25/05/09	1/06/09	1W
2	Literature review	2/06/09	2/07/09	4W
3	Analysis And Design	3/07/09	24/07/09	3W
4	Implementation and testing	25/07/09	3/09/09	6W
5	Writing up and Submission	25/05/09	3/09/09	15W

Table 1: Project Plan

1-6- Resource of the Project

In this project, we used academic paper, books and internet to complete my literature review on this subject area like "Computer Security" for Matt Bishop and supervisor PHD thesis. ASP.net and SQL 2005 software were used to implement the project.

1-7-Tools and Technologies Used

Policy Editor for SANTA is a basic user-friendly system to facilitate the non –specialist users to perform the intended activities. System has been developed on the Microsoft Platform using the following tools and technologies.

- Visual Studio 2008 as development tool.
- Microsoft .Net Framework 3.5 as development framework.
- C# as a language.
- SQL Server Management Studio as Database Tool.
- SQL Server 2005 as Database Server.
- Rational Rose as Designing Tool.

1-8 Dissertation Outline

In Chapter 2, I provide a comprehensive literature review about security requirements and security policy. It provided the definition of a security policy and security requirements. In addition I give an overview of policy models and policy languages. In the final part of the chapter, I present security threats.

In Chapter 3, I identify relevant components of the SANTA policy language. SANTA is a language for specifying security policies such as authorization and obligation policies.

In Chapter 4, I present the Finite State Machine model of the system, designed using JFLAB tools.

In Chapter 5, I analyze and design a user-friendly web-based policy editor. This allows non specialist users (normal users) to create and manage security policies in SANTA.

In Chapter 6, I implement a user-friendly web-based policy editor that allows non specialist users (normal users) to create and manage security policies in SANTA.

In Chapter 7, I test and evaluate the system as whole.

In Chapter 8; the final chapter, I propose conclusions of this dissertation and also provide recommendations for future work on the topic.

Chapter 2: Literature Review

2-1- Overview

This chapter presents an overview of some significant writings on security requirements and security policy.

2-2- Security Requirements

The main objective of the security requirements are to safe-guard the information data against potential risks. They usually provided as a mean of permitting access to an organization's system resources, whilst setting out a user guidelines for various individuals who have permission to access the system and their limitations of use [1].

2-3- The fundamental Components

The integral components of computer security comprises of confidentiality, integrity and availability. These features are different in their characterization from one another. They also vary in the framework each aspect is applied. However, individual requirements, customs and laws of a specific organization determine the interpretation of the aspect in that particular context or environment [1].

2-3-1- Confidentiality

Confidentiality is defined as "the concealment of information or resources" [1]. The application of computer in sensitive environments (for e.g. government and industry) has necessitated the undisclosed use of information. For instance, revealing the exact statistic of people who have doubts about politicians is not as significance as revealing that the poll itself was carried out by a subordinate of the politicians, hence concealing resources is a feature of confidentiality. Several website hide their application system and configuration as well as organization may conceal certain equipment they use [1].

2-3-2- Integrity

Integrity defines the reliability of the data assets. It is usually designed in such a way to avoid misuse of an organization's data resources or unofficial alterations to the system. Usually, integrity incorporates both data (the content of the information) and origin (the data source) integrity. The latter is commonly known as authentication. The individual user rating and confidence on the information is the underlying measure for the accuracy and credibility of the source of the information. "This dichotomy illustrates the principle that the aspect of integrity known as credibility is central to the proper functioning of a system" [1].

Integrity mechanisms fall into two classes:

2-3-2-1- Prevention Mechanisms

This method makes sure unauthorized efforts to alter data resources are prevented so as to retain the integrity of the information. The process also blocks any activity that can lead to alterations of the data resources in an unauthorized manner. However, it's important to understand the difference between the two forms of attempts. The first attempt is applicable when an individual who has no official permission attempts to change information, whereas the second one happens to deny permitted user from executing certain alteration beyond his jurisdiction [1].

2-3-2-2- Detection Mechanisms

The detection mechanisms basically report when the information integrity cannot be trusted any more. Unlike the prevention mechanisms, the detection mechanisms do not attempt to block violation of integrity. However, it examines sequence of the system events in order to identify the problem. At the same time, it may check the data information to detect whether required or expected restrictions are still in place. These mechanisms could give detail

of the possible cause of the integrity violation. In some cases declares the file for being corrupted [1].

2-3-3- Availability

Availability of required resource or information is its ability to be used. This is a significant requirement of reliability and system design, as inaccessible system is no different to lack of the system in any ways. The organization may decide to make the data unavailable or consciously prevent the information access which is an aspect of availability. This phenomenon is vital to maintaining the computer security. The designed system normally assumes a statistic model in order probe expected pattern of use. This is vital to availability, as it guarantees availability of the data when specifications of the statistical model are met. However, an individual could be capable to influence the access, hence rendering invalid assumptions by the model. This therefore, causes failure of the system, as the method allowing availability of the data resources are operating outside its prescribed context [1].

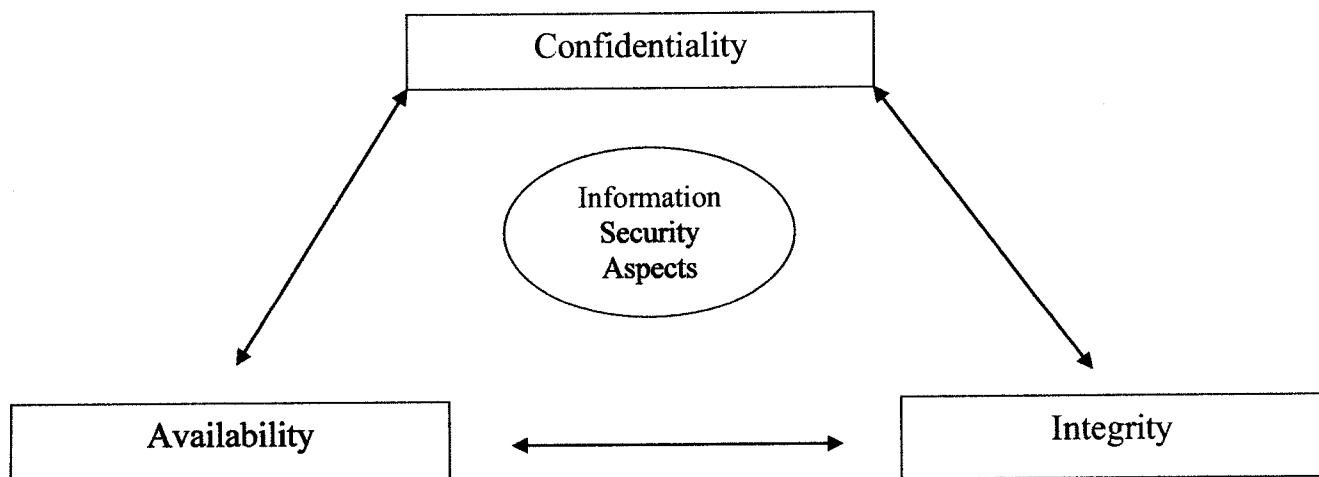


Figure 1: Information Security Aspects (adopted from: Russell and Gangemi, 1991)[12].

2-4- Security Policy

2-4-1- Definition

A security policy defines specified rules and regulation set to ensure communication is maintained between the staff managing the system and individual user. Security policy is expected to interpret and verify primary security. Hence security policy acts as link between the management's objective and the security requirement of the users [6].

2-4-2- Security Policy Models

2-4-2-1- Overview

A company's security policy models summarize statements of how organizational assets and properties are to be protracted, typically written in one page or less. Organizations typically state their protection goals of various systems as agreed by the senior executive management and hence functions as the basis of guidance of do and don'ts with regards to protection and maintenance of systems, assets and other sensitive organizational properties that are susceptible to compromise. There are different security models adopted by different organizations, each has merits and demerits and sometimes related to protection of assets on a specific industry. Below is a summary of some for the leading and popular models in the security policy [4].

2-4-2-2- Bell-LaPadula Policy Model

This model was adopted as a result of a research work conducted by David Elliott Bell and Leonard J. La Padula in the 1970s. This security model is an access control that uses formal and mathematical description approach to explicitly denote how a security should be maintained, by using some sort of comparison to determine an access right based on the authorization level of the requested and the level of access authorized on the object. There was a widespread realization that protection of commercial operating systems was

poor, bugs and system vulnerabilities kept being discovered on commercial systems as soon as one is fixed. As a result, US government introduces a reference monitor concept and this was adopted as a result of a study conducted by James Anderson who concluded that protection of secure systems and properties should be enforced by using a simplified verifiable mechanism that rarely change. Reference monitor is part of Trusted Computing Base (TCB). TCB is a set interrelated hardware, software human components that together ensures enforcement of a security policy failure of which causes a breach of security policy [6].

2-4-2-3- Clark-Wilson Model

This security policy model is widely used in commercial environments to primarily ensure integrity of organizational data and prevent corruption of data due to error or malicious intent. It was proposed by David D. Clark and David R. Wilson in 1987. The policy clearly defines how organizational data should be kept securely with data integrity intact. It distils centuries old double-entry bookkeeping method which tends to mitigate insider fraud and ensure integrity of bank's accounting system. By posting each transaction to two different books namely Credit and Debit (assets and liabilities respectively), the accounting book should balance anytime a reconciliation is done. This ensures information integrity. Enforcing this policy entails principle of splitting job responsibilities to more than one staff and hence any potential fraud may requires collusion of two or more staff [6].

2-4-2-4- Chinese Wall Model

Is a popular security model proposed by Brewer and Nash in 1989. This policy tends to prevent flow of information in situation where conflict of interest is prevalent. This policy is widely used by professional consultants and investment bankers to build a virtual wall that separates the control the flow of information in order to avoid a conflict of interest or insider trading in a financial setup. For example, a conflict of interest arises here of an individual consults for two companies of same industry, particularly in a financial sector, as soon as a consultant interacted with two companies of same time business line. A closed 'Chinese Wall' is erected once an individual consulted for companies of same industry type and hence avoids conflict of interest on the job [6].

2-5- Security Policy language

2-5-1- Overview

This is the language that represents the security policy. The security policy languages are categorized into two groups; High-level policy language and Low-policy language.

2-5-2- High-Level Policy Languages

These refer to the restrictions imposed on the system's entities and commands which provide defined terms of the security policy. The accuracy of such expression demands formulation mathematical model of the security where ordinary English could not be applied [1].

2-5-3- Low-Level Policy Languages

A low-level policy language refers to a collection of inputs, argument to command those inputs and to verify the limitations on the system [1].

2-6-Policy Management

Customary reviews have to be employed in order to make sure the security policy is up to date. This may be executed in such a way that the organization's operation systems are directly translated into the security policy.

It was suggested that a special unit such as data security unit be assigned to administer the security policy. The unit would be in charge of performing expected reviews and keeping the security policy up to dated [6].

Killmeyer (2006) outlined the duty of security management. He stated the security management should provide direction and put procedures in place to control and minimize the risk of losing data and information. He also stressed that the security management requires further security measures in order to identify the origin of all message coming through and their authenticity. When the messages are strictly authenticated, that will help to reinforce security, and stop any attempt to forget identities [10].

2-7-Repudiation

Killmeyer also discusses the concept of repudiation. Non repudiation means receiving a proof of delivery and certification of the source. This is a method which guarantees that the sender cannot later deny that he/she has sent the message and also the recipient cannot refuse that he/she has received the message. This is an alternative way to manage and apply security by monitoring the messages going through the networking system[10].

2-8-Development of security policy

Security policy is significant in ensuring efficient and understandable security systems are created. Although the essential nature of developing security policy is frequently ignored, it serves as a vital element of the computer security design. The main purpose of the security policy is to decode the management's requirements for the security into a simple well defined and directed to a precise targets or objectives. Using a top down method is essential to derive an efficient and well developed overall security architecture. In the other hand, lack of security policy to relay the management's expectations will result in such decisions being made directly by individual users. Therefore significant operations such as installation and maintaining the computer system will be left in the hands of the individual users to be made. These consequently would lead to the implementation of an ineffective computer security systems or architecture [6].

2-9-Security policy violation

Violation of the security policy is a very serious threat to an organization's data assets. The violation may not necessarily take place in order to pose a risk at an organization's security resources. However it's essential to protect against those activities that may lead to the security violation. The actions that cause violation are commonly known as attacks which are usually carried out by attackers [2].

2-9-1-Interception

Interception is an unlawful disclosure of information which is a serious breach and a risk to confidentiality. This is mostly carried out through passive wiretapping which happens when listening to certain entity, reading or browsing files or systems information. The threat is also referred to as snooping or eavesdropping [2].

2-9-2-Modification/Alteration:

This is an unlawful alteration of the data security assets which is a threat to the data reliability. As opposed to interception, modification is an active form of violation. This might be achieved by active wiretapping, which is an alteration that influences the nature of data communication between networks. A typical example is the man in middle attack where an intruder monitors and reads information from sender and then sends a modified edition to the recipient. In most cases, the presence of a third-party is not felt by either the sender or the reader [2].

2-9-3- Masquerading/Spoofing

Another threat to the data security reliability includes the violation such as masquerading or spoofing, which involves masquerade of an individual by another person. In this case, the violators disguise the true identity of the service from the individuals at the receiving end so as to entice him accept a particular entity [2].

2-9-4- Interruption

Threats to the data availability include a form of infringements such as “interruption”. This mainly takes place in a form of an unlawful denial of users’ access into specific data resource. A common example is revealed when an intruder influences the server

performing its service; a breach known as the denial-of-service attack. The attackers accomplish this through various mechanisms such as denial at the source which stops the server from receiving the information supply. These data resources are required for effective function of the server at its local environment thus hindering information transmission via the server across networks. The denial may also occur on the route of the intermediate pathway thereby getting rid of the information from the server or the client or influencing both of them [2].

2-9-5-Fabrications

Fabrications are another form of violation which compromises the security integrity. These involve assembly of forged resources or unlawful addition of information. A common example is the insertion of false information in a network [2]. However, the staffs are required to develop complete understanding and knowledge of the costs of the policy violation, so as to be able to appreciate the significance of the security policy.

Security policy violation is a serious threat to an organization's corporate systems and should be dealt appropriately. Staff or individuals who violated the policy must be notified about their action and cautioned to face penalty. These may be done by placing them on a "trial period" by allowing them access only limited resources until they demonstrate competency to utilize the organization information in a safe way and comply with the security policy when using the corporate systems. In a more serious case of violation, the violator should be warned of being sacked or getting prosecuted. Although the latter punishment may seem to be too much, reasonable step has to be taken in any case of violation following the terms or guidelines as outlined in the AUP and the policy. However, the main concentration should be on maintaining the security rather than just the punishment for the violation. Without this, it would be difficult to penetrate which might be as a result of human mistake or misinterpretation of the security policy [5].

Chapter 3: SANTA Language

3-1- Overview

The chapter introduces a security policy language commonly known as the SANTA. We would also look into the policy in SANTA by unfolding the authorization, obligation and delegation.

3-2- Definition

SANTA (Security Analysis Toolkit for Agent) is language for specifying security policies such as authorisation and obligation policies. SANTA is developed in the Software Technology Research Laboratory (STRL) at DMU and is used to specify and analyse security policies using a run-time verification tool called Tempura.

3-3-Policies in SANTA

A security policy in SANTA is based on the notion of subject, object and action. A subject is assumed to be an active entity that can perform an action on an object or another subject. Examples of subject are users, processes or agents running on behalf of users. An object is a passive entity that stores some data, e.g. a file, a storage device or a profile. An action is an operation that can be performed on objects or subjects, e.g. read a file, append to a file, withdraw money from a bank account and so on.

3-4- Syntax of SANTA Policy

3-4-1- Simple Condition

The syntax of the simple condition is that

<code>< cond > ::= < exp1 > < rop > < exp2 ></code>	relational conditions
<code>< rop > ::= = > < ≤ ≥</code>	relational operators
<code>< exp > ::= < var ></code>	variable symbols
<code> < const ></code>	constant symbols such as true, false, 0, 1, 2
<code> - < exp ></code>	unary minus
<code> (< exp >)</code>	parentheses
<code> < exp1 > < aop > < exp2 ></code>	binary arithmetic operations
<code>< aop > ::= + - * /</code>	arithmetic operators

Example of simple conditions:

- $x = 5$
- $x + y > -1$
- $2 * (x + 3) \leq 0$

3-4-2- Compound Condition

The syntax of the Compound condition is that

$\langle \text{Ccond} \rangle ::= \langle \text{cond} \rangle$	simple condition
$ \langle \text{Ccond1} \rangle \wedge \langle \text{Ccond2} \rangle$	conjunction
$ \langle \text{Ccond1} \rangle \vee \langle \text{Ccond2} \rangle$	disjunction
$ \neg \langle \text{Ccond} \rangle$	negation
$ (\langle \text{Ccond} \rangle)$	parentheses

Example of compound conditions:

- $x = 5$
- $(x + y > -1) \wedge \neg (x \leq 0)$

3-4-3- Policy Rule

3-4-3-1- Authorization Rule

This rule specifies access control measures. Authorization involves three types of rules namely positive authorization, negative authorization and decision rules[3].

Positive Authorization

These are declarations that express specific conditions under which specific summon for access could be tolerated. However, as far as ultimate decision of the policy for the right of system entry is concerned, only the signal is the taken into consideration [3].

A positive authorization rule has the form

$$f \longrightarrow \text{autho}^+(X, Y, Z)$$

The letter f represents ITL formula in this syntax. X,Y,Z represent subject, object and action respectively. The rule expresses that the subject X is clearly granted to carry out

A web-based policy editor for SANTA	العنوان:
Al Mutairi, Abd Algder	المؤلف الرئيسي:
Siewe, Francois(Supervisor)	مؤلفين آخرين:
2009	التاريخ الميلادي:
Leicester	موقع:
1 - 105	الصفحات:
687206	رقم MD:
رسائل جامعية	نوع المحتوى:
English	اللغة:
رسالة ماجستير	الدرجة العلمية:
De Montfort University	الجامعة:
Faculty of Computing Sciences and Engineering	الكلية:
بريطانيا	الدولة:
Dissertations	قواعد المعلومات:
الأمن المعلوماتي، هندسة البرمجيات، الإنترنت، لغة البرامج	مواضيع:
https://search.mandumah.com/Record/687206	رابط:

References

- 1- Bishop, M, (2003), "*Computer Security Art and Science*", Addison- Wesley.
- 2- Siewe, F (2005), "*A Compositional Framework for the Development of Secure Access Control System*", PHD Thesis, DMU university.
- 3- Janick, H, (2007), "*The Development of Secure Multi-Agent System*" PHD Thesis, DMU university.
- 4- Anderson, R and Stajano, F, "*Security Policies*" ,Paper , Available online
<http://www.cl.cam.ac.uk/~rja14/Papers/security-policies.pdf>
- 5- Danchev, D, (2003), "*Building and Implementing A successful Information Security Policy*" Paper, Available online
http://www.infosecwriters.com/text_resources/pdf/Security_Policy_JPak.pdf
- 6- Weise, J, (2001) "*Developing A security Policy*" Available online
<http://www.sun.com/blueprints/1201/secpolicy.pdf>
- 7- Bilung, Lee and Edward, A. Lee "*Interaction of Finite State Machines and Concurrency Models*", University of California at Berkeley.
- 8- Susan H, Rodger, Jinghui Lim, Stephen Reading "*Increasing Interaction and Support in the Formal Languages and Automata Theory Course*".
- 9- <http://www.tech.dmu.ac.uk/STRL/research/software/index.html#SANTA>
- 10- Kilmeyer, J. (2006) "*Information Security Architecture An Integrated Approach to Security in the Organization*". Boca Raton New York: Auerbach Publication.
- 11- Sindell, K.(2002), "*Safety Net Protecting Your Business On the Internet*". New York: John Wiley & Sons, Inc.

12- Russell, D. and Gangemi, G. T. (1991), "*Computer Security Basics*".

Cambridge: O'Reilly & Associates, Inc.

13- Sommerville, I. (2007). "*Software Engineering*".

14- Lano et al. (2002,25). "*Design Process*".

15- <http://www.issco.unige.ch/en/research/projects/ewg95//node108.html>.

A web-based policy editor for SANTA	العنوان:
Al Mutairi, Abd Algder	المؤلف الرئيسي:
Siewe, Francois(Supervisor)	مؤلفين آخرين:
2009	التاريخ الميلادي:
Leicester	موقع:
1 - 105	الصفحات:
687206	رقم MD:
رسائل جامعية	نوع المحتوى:
English	اللغة:
رسالة ماجستير	الدرجة العلمية:
De Montfort University	الجامعة:
Faculty of Computing Sciences and Engineering	الكلية:
بريطانيا	الدولة:
Dissertations	قواعد المعلومات:
الأمن المعلوماتي، هندسة البرمجيات، الإنترنت، لغة البرامج	مواضيع:
https://search.mandumah.com/Record/687206	رابط:

A Web-based Policy Editor for SANTA

MSC SOFTWARE ENGINEERING
COMP5314

Dissertation

SUPERVISED BY Francois Siewe
Done by Abdugder Almutairi

P06005561

2009
DeMontfort
university