

# **Palestine Polytechnic University**



College of Engineering & Technology

Electrical and Computer Engineering Department

Graduation Project

## **JCOP: A Security Framework for JADE Intra Platform Mobility**

### **Project Team**

Salman Qabaga

Khaliel Ikhlaiel

Saqer Atawneh

### **Project Supervisor**

**Dr. Radwan Tahboub**

Hebron- Palestine

2011

جامعة بوليتكنك فلسطين

الخليل – فلسطين

كلية الهندسة و التكنولوجيا

دائرة الهندسة الكهربائية و الحاسوب

اسم المشروع

**JCOP A Security Framework for JADE Intra Platform Mobility  
Over windows Operating System**

أسماء الطلبة

صقر عطاونة

خليل اخليل

سلمان قباجة

بناء على نظام كلية الهندسة و التكنولوجيا و إشراف و متابعة المشرف المباشر على المشروع  
وموافقة أعضاء اللجنة الممتحنة تم تقديم هذا المشروع إلى دائرة الهندسة الكهربائية و الحاسوب  
وذلك للوفاء بمتطلبات درجة البكالوريوس في الهندسة تخصص هندسة أنظمة الحاسوب.

توقيع المشرف

.....

توقيع اللجنة الممتحنة

.....

.....

.....

توقيع رئيس الدائرة

.....

## **Abstract**

Mobile agent technology offers a new computing paradigm in which a program, in the form of a software agent, can suspend its execution on a host computer, transfer itself to another agent-enabled host on the network, and resume execution on the new host.

The JCOP is a security framework for JADE intra platform mobility; it consists of a set of agents that cooperate with each other to provide security for the platform, and a set of classes for programmers to integrate their work with JCOP access policy. This project aims to enforce security over JADE with the minimum amount of performance overload.

## **Acknowledgments**

*First of all thanks for God that make us go that far in this project and for the successes we reach.*

*For our families' good friends, for our advisers Dr. Radwan Tahbub and Dr. Amal Aldweak Wazwaz and for Palestine Polytechnic University we dedicate this work.*

## Table of contents

Subject	Page
<b>Cover page</b> .....	<b>I</b>
<b>Signature page</b> .....	<b>II</b>
<b>Abstract</b> .....	<b>III</b>
<b>Dedication</b> .....	<b>IV</b>
<b>Acknowledgments</b> .....	<b>V</b>
<b>Chapter One : Introduction</b> .....	<b>1</b>
1.1 Overview .....	<b>2</b>
1.2 Project Important .....	<b>3</b>
1.3 Project Objectives .....	<b>3</b>
1.4 Literature Review .....	<b>4</b>
1.5 Requirements .....	<b>6</b>
1.5 Time Plane .....	<b>8</b>
1.6 Road Map .....	<b>9</b>
<b>Chapter Two : Theoretical Background</b> .....	<b>11</b>
1.2 Distributed System .....	<b>12</b>
• Definition Of A Distributed System .....	<b>12</b>
• Distributed System Goals .....	<b>12</b>
• Types of Distributed Systems .....	<b>13</b>
• Communication of Distributed System .....	<b>14</b>
• Distributed System Security .....	<b>14</b>
2.2. Mobile agent .....	<b>15</b>
• Definition of an agent system .....	<b>15</b>
• Definition of Mobile agent .....	<b>16</b>
• Mobile agent Components .....	<b>17</b>

• Mobile agent life cycle .....	18
• Mobile agent security requirements .....	19
2.3 Security Issues in Mobile Agent .....	20
Agent –to- platform threats .....	23
Agent –to- Agent threats .....	24
Platform –to- Agent threat .....	25
Other –to- agent platform threats .....	26
2.4 Software Description .....	28
2.4.1. The Jade Platform .....	28
2.4.2. Java programming language .....	30
2.4.3. NetBeans IDE .....	31
<b>Chapter Three: System conceptual Design .....</b>	<b>32</b>
3.1. detailed project objectives .....	33
3.2. Security Solutions .....	34
• System Definition .....	34
• System block diagram .....	34
• Problem Solving .....	37
• System Modeling .....	38
3.3 Testing Application .....	41
3.3.1 System Definition .....	41
3.3.2 System Block Diagram .....	42
3.3.3 System Modeling .....	45
<b>Chapter Four : Detailed System Design .....</b>	<b>47</b>

4.1 Security Manager (SM) Architecture	48
4.2 Detailed Specification of SM Components .	
4.2.1 Registration Authority (RA)	50
4.2.2 Certification Authority (CA)	50
4.2.3 Validation Authority	51
4.2.4 Container Guardian	51
4.2.5 WatchManV1	51
4.2.6 WatchManV2	51
4.3 Agents Task Scheduling and Interactions in JADE	52
4.3.1 Agent Task in JADE	53
4.3.2 Primary types of behaviors in jade	53
4.3.3 Scheduling operation	54
4.3.4 Methods invoked during the agent life Cycle	55
4.3.5 Interacting with AMS	56
4.4 Detailed System Modeling	56
4.4.1 Security Manager Agents	57
4.4 .1.1 Registration Authority (RA)	57
4.4 .1.2 Certification Authority (CA)	63

4.4 .1.3 Validation Authority (RA)	.....	<b>66</b>
4.5 Description of interaction classes	.....	71
4.7 Limitations	.....	77
4.8 Assumptions	.....	77
4.9 Summery	.....	78
<b>Chapter Five :Implementation and Testing</b>	.....	<b>79</b>
5.1 Development Environment		80
5.2 Development Process		81
5.3 Testing		82
5.4 Summary		99
<b>Chapter Six :Conclusion and Future Work</b>	.....	<b>100</b>
6.1 experimental Results		101
6.2 Conclusion		106
6.3 Future Work		106
6.4 Summery		106



## Table of Figures

Figure	Page
Figure 2.1. Distributed System Model .....	13
Figure 2.2. The Mobile Agent Paradigm.....	17
Figure 2.3. Mobile Agent Life Cycle.....	19
Figure 2.4. Simple mobile agent s system.....	21
Figure 2.5.Main architectural elements of a JADE platform .....	29
Figure 2.6. Relationship between the main architectural elements.....	29
Figure 3.1 System Block Diagram.....	35
Figure 3.2 Security Manager Block diagram .....	36
Figure 3.3 Flowchart Describes RA Agent Behavior.....	38
Figure 3.4 Flowchart Describes The CA Agent Behavior .....	39
Figure 3.5 Flowchart Describes The VA Agent Behavior.....	40
Figure 3.6 General idea of the FNS.....	41
Figure 3.7: System Block Diagram.....	42
Figure 3.8: First Strategy of Mobile Agent in FNS.....	43
Figure 3.9: Second Strategy of Mobile Agent in FNS.....	44
Figure 3.10:Third Strategy of Mobile Agent in FNS.....	44
Figure 3.11: Show refined dataflow – level 0 which gives details for the searching process to explicitly describe each subprocess.	45
Figure 3.12 Dataflow diagram – level 1.....	46

Figure 3.13 Use case to FNS.....	46
Figure 4.1 Architecture of SM.....	50
Figure 4.2. Agent Thread Path Of Execution.....	53
Figure 4.3: RA Global Behavior.....	59
Figure 4.4 .RA Step (1).....	60
Figure 4.5. RA Step (2) .....	61
Figure 4.6 .RA Step (3) .....	63
Figure 4.7. RA Step (4) .....	64
Figure 4.8 .RA Step (5) .....	65
Figure 4.9 .CA Global Flowchart.....	66
Figure 4.10: CA block A and B.....	67
Figure 4.11 .CA block C, D, E, and F.....	69
Figure 4.12 The VA Global.....	70
Figure 4.13: VA Block A And B.....	71
Figure 4.14 VA Block D And E.....	72
Figure 4.15 WMV1 Flowchart.....	73
Figure 4.16 WMV2 Flowchart.....	74
Figure 4.17 Guardian Flowchart.....	76
Figure 4.18 FNS Behaviors.....	77
Figure 4.19FNS Flow Chart.....	78

Figure 5.1 WMV1 show the name rma after discovering it.....	84
Figure 5.2 WMV1 show the name df after discovering it.....	85
Figure 5.3 WMV1 show the name RA after discovering it. ....	85
Figure 5.4 WMV1 show the name ams after discovering it.	86
Figure 5.6 WMV1 show the name of new agent called “agent” after discovering it.	86
Figure 5.6 RA show content of a message sent by WMV1 carrying the name of the new agent (this agent did not follow the JCOP restrictions).	87
Figure 5.7 RA kills the new agent “agent”. ....	87
Figure 5.8 WMV1 show the name of new agent called “agent1” after discovering it. ....	88
Figure 5.9 RA show content of a message sent by WMV1 carrying the name of the new agent (this agent follow the JCOP restrictions but with illegal goal). ....	89
Figure 5.10 a message box that shows the content of agent request message (notice the word kill in the message content where kill is restricted word)	90
Figure 5.11 a message box shows the request message inside the text validation process. ....	91
Figure 5.12 RA kills the new agent “agent1” because it has illegal goal.....	91
Figure 5.13 WMV1 show the name of new agent called “agent2” after discovering it.	92
Figure 5.14 RA show content of a message sent by WMV1 carrying the name of the new agent (this agent follow the JCOP restrictions). ....	93
Figure 5.15 a message box that shows the content of agent request message (notice that the message did not contain any restricted word) ....	94
Figure 5.16 a message box shows the certification granted from CA to new agent. ....	95

Figure 5.17 the JCOP certStore and keyStore directories. ....	96
Figure 5.18 inside JCOP certStore directory. ....	96
Figure 5.19 inside JCOP Certificate0 directory. ....	97
Figure 5.20 inside JCOP Certificate0 file. ....	97
Figure 5.21 inside JCOP RequestText0 file. ....	98
Figure 5.22 inside JCOP Signature0 file. ....	98
Figure 5.23 the FNS stationary agent “Controller”. ....	99
Figure 5.24 the Agent0 is the searcher mobile agent. ....	99
Figure 5.25 the searcher agent search in machine 3. ....	100
Figure 5.22 the searcher agent search in machine 1. ....	101

## **List of tables**

Table 1.1. Tasks description (1 <sup>st</sup> semester)	<b>8</b>
Table 1.2 Time Plane (1 <sup>st</sup> semester)	<b>9</b>
Tabel 4.1 :RA Step(1)	<b>60</b>
Table 4.2. RA Step (2)	<b>61</b>
Table 4.3: RA Step (3)	<b>62</b>
Table 4.4. RA Step (4)	<b>64</b>
Table 4.5. RA Step (5)	<b>65</b>
Table 4.6: CA block A AND B	<b>67</b>
Table 4.7: CA Block C , D, E, F	<b>68</b>
Table 4.8: VA Block A,B and C	<b>70</b>
Table 4.9. VA Block D and E	<b>71</b>
Table 4.10 WMV1	<b>73</b>
Table 4.11 WMV2	<b>74</b>
Table 4.12 Guardian	<b>75</b>
Table 6.1 Hardware used in the experiments	<b>103</b>
Table 6.2 Experiment 1 Memory Comparison	<b>105</b>
Table 6.2 Experiment 1 CPU Comparison	<b>105</b>
Table 6.3 Experiment 2 Memory Comparison	<b>106</b>

Table 6.4 Experiment 2 CPU Comparison

Table 6.5 Experiment 3 Memory Comparison **107**

Table 6.6 Experiment 4 CPU Comparison **107**

# **Chapter 1**

## **Introduction**

- 1.1 Overview**
- 1.2 Project Importance**
- 1.3 Project Objectives**
- 1.4 Literature Review**
- 1.5 Requirements**
- 1.6 Time Plan**
- 1.7 Road Map**

# **Chapter One**

## **Introduction**

### **1.1. Overview**

Through last two decades many complex methods were found to support the communication over distributed systems such as RMI, DCOM, and CORBA [1]. These methods had disadvantages; it was very expensive, more suitable for large systems, and increase the network load and latency time in communication over distributed systems. So, many problems with different aspects where needed to be analyzed and put a solution for it.

One of many methods found to communicate over distributed systems is the mobile agent paradigm. According to GRAY [GKRNC96] a software agent is “a program that is autonomous enough to act independently, even when the user or application that launched it is not available to provide guidance and handle errors “. In another definition, using different terms, a software agent is program that acts in behalf of its owner (agent owner) [2].

A mobile software agent, or a software agent for now on, is an object that migrates through many nodes of a heterogeneous network of components, under its own control, in order to perform tasks using resources of these nodes [2]. It travels from node to another of a distributed system performing tasks on behalf of its owner. At the end of this process an agent can return to its home site and report itself to the user who injected this object in the distributed system .The mobile agents simply provides a greater opportunity for abuse, misuse, and broadening the scale of threats significantly [2].



In this project we will discuss approaches of mobile agent security and implement one of these approaches in JADE platform to provide a secure mobile agent.

## **1.2. Project Importance**

The adoption of the Mobile agent's technology is currently limited by the lack of security. Mobile agents face large scale of threats that act on mobile agent or on its platform. These threats can affect mobile agent themselves or on data carried by it.

So, it is important to secure mobile agent against those threats to encourage users and companies to use the mobile agent. This improves the effectiveness of network and decreases the network latency; therefore the secure mobile agent that guarantees authentication, confidentiality and integrity can travel over distributed system safely.

## **1.3. Objectives**

- Making survey about the mobile agent common security threats (vulnerabilities).
- Search for acceptable solutions for these security threats.
- Implement these solutions with JADE development environment.
- Implement the simple file name searcher as JADE multi\_agent application.
- Apply the security solutions to the FNS application to test their effect.

## **1.4. Literature Review**

### **1.4.1. Advanced mobile agent security models for code integrity and malicious availability check:**

- **Objective:** Malicious Identification Police model [MIP] for scanning the incoming agent to detect the malicious activities and to overcome the availability of vulnerabilities in the existing Root Canal algorithm for code integrity checks.
- **Work done:** The MIP model is extended with the policy to differentiate the agent owners in the distributed environment and the Root Canal algorithm is improved as extended Root Canal algorithm.
- **Conclusion:** The experimental results of the advanced models show that though these mechanisms take more time complexity than the existing malicious identification police model and Root Canal model, these models are efficient in protecting the agent code integrity and scanning the agent for malicious activities. Also the new models possess less time complexity compared to the other related existing models in the secure mobile agent environment [3].

### **1.4.2. A buddy model of security for mobile agent communities operating in pervasive scenarios:**

- **Objective:** every agent protects its neighbor within the community, thereby sharing the responsibilities of the security function. This feature akas it a better option as compared to other hierarchical models of security, which can be brought down by a concerted attack at the controller agent.

- **Work done:** The buddy model of security for mobile agent.
- **Conclusion:** This study also demonstrates the applicability and the effectiveness of the Buddy model in different pervasive scenarios and makes a strong case for its adoption [4].

#### 1.4.3. A security protocol for mobile agents based upon the cooperation of sedentary agents:

- **Objectives:**
  - ✓ The cooperation between a mobile agent and a sedentary agent.
  - ✓ The reference execution (reliable platforms which shelter our cooperating sedentary agents).
  - ✓ The use of cryptography and the digital signature to ensure safe inter-agent communication and time-limited execution (timeout).
- **Work done:** produce dynamic approach which makes use of a timer to make it possible to detect a mobile agent's code re-execution was used.
- **Conclusion:** The attack on agent permanent modification was also dealt with. Moreover, the protocol is sufficiently robust so that it is durable and fault tolerant [5].

#### 1.4.4. Verifiable Distributed Oblivious Transfer (VDOT):

- **Objective:** Using VDOT scheme allows us to replace a single trusted party with a group of threshold trusted servers.
- **Work done:** The design of VDOT uses a novel technique called consistency verification of encrypted secret shares.

- **Conclusion:** VDOT protects the privacy of both the sender and the receiver against malicious attacks of the servers. The preliminary evaluation shows that protecting mobile agents not only is possible, but also can be implemented efficiently [6].

#### **1.4.5. Securing mobile agent using multi agent cryptographic protocols:**

- **Objective:** Using this scheme will eliminate the need of trusted third party, and provide protective protocols to secure mobile agents most real world attacks.
- **Conclusion:** These protocols heavily rely on well-established cryptographic primitives, such as encrypted circuits, threshold decryption, and oblivious transfer [7].

### **1.5 Requirements**

Since this project is divided into two parts. The Requirements of the two parts will be stated separately

#### **1.5.1 System security Requirements**

To protect the mobile agent system then the two main components of the system must be protected: agent and its platform.

To achieve this protection, set of procedures are defined:

- A Registration Authority (RA) must starts when JADE main Container lunch. To verify agents goal and allow agents to legally register in JADE.

- Trusted Certificate Authority (CA), must start when the JADE main container lunch. To certify the legally verified agents by RA.
- A Validation Authority must starts when JADE main container lunch. To verify and validate agents movements in the intra connected platform.
- Introspector to inform the RA when new agent is created.
- Introspector to inform the VA when agent moves from container to another.
- Certification Store to store the certificates issued by CA.
- Key Store to store the public Key produced by CA.
- Class that encapsulate the certificate data inside the agent.
- Class that allow programmers to integrate the security mechanism to their applications.

### **1.5.2 Requirements of testing application (File Name Searcher)**

#### **1.5.2.1 User Requirements**

The system should provide a graphical user interface (GUI) in order to interact with the user in an easy and simple way. The GUI should provide:

- Option for the user to provide the system with the file name.
- Option for the user to start the search process.
- Option for user to stop the search process.
- Option gives user the search result.

#### **1.5.2.2 Functional Requirements**

This System is composed of two Stationary Agents: agent called reference agent (RFA) and mobile agent, called Searcher Agent (SA).

When the application gets the file name form user it provides it to the RA which creates SA that caring the file name. Reference Agent sends the Searcher Agent to search for target over a network. When Searcher Agent finds the file then it will send a message with the result to Reference Agent. In turn, Reference Agent sends kill message to the Searcher Agent and finish the search process.

### 1.5.2.3 NON Functional Requirements

- ❖ Usability, reliability, security, and performance should be provided as follow
  - Usability: the system should be easy to use.
  - Reliability: the system should be reliable under different circumstances.
  - Security: communication between agents should be secure.
  - Performance: the system should work with good performance measures.

## 1.6. Time Plan

**Table 1.1. Tasks description (1<sup>st</sup> semester)**

Task ID	Task Description
<b>T1.1</b>	Selecting the project.
<b>T1.2</b>	Collecting information.
<b>T1.3</b>	System and requirements analysis.
<b>T1.4</b>	System design.
<b>T1.5</b>	Documentation.

**Table 1.2 Time Plan (1<sup>st</sup> semester)**

Task/Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
<b>T1.1</b>																
<b>T1.2</b>																
<b>T1.3</b>																
<b>T1.4</b>																
<b>T1.5</b>																

**Table 1.3 Tasks description (2nd semester)**

Task ID	Task Description
<b>T1.1</b>	System design.
<b>T1.2</b>	Learn JADE
<b>T1.3</b>	Learn Java security API
<b>T1.4</b>	Implement Security solution (JCOP)
<b>T1.5</b>	Documentation.

**Table 1.4 Time Plan (2nd semester)**

Task/Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
<b>T1.1</b>																
<b>T1.2</b>																
<b>T1.3</b>																
<b>T1.4</b>																
<b>T1.5</b>																

## 1.7. Road Map

This project contains six chapters, each one describes a specific task of the project, and this page is the finale page in chapter one, it consist of briefly description about next five chapters.

**Chapter Two:** Theoretical Background, this chapter gives a clear picture about the system theoretical background related to the main concepts of agent platform, agent, mobile agents, mobile agent security, and system basic components.

**Chapter Three:** Conceptual design, this chapter shows system block diagram, design options that can be used in the system, and each system component interface with each other.

**Chapter Four:** Detailed design, this chapter describe the design of the system including flowcharts, AUML, and GUI design.

**Chapter Five:** System implementation and testing, this chapter describes the system implementation and testing in details.

**Chapter Six:** Conclusion and future work, this chapter gives how much we are achieved from our objectives, what is the problems we face and how we achieve the solution, and finally we will give suggestions for the future development.



# **Chapter Two**

## **Theoretical Background**

**2.1 Distributed Systems**

**2.2 Mobile Agents**

**2.3 security issues**

**2.4 Software Description**

## **Chapter Two**

### **Theoretical Background**

This chapter provides an illustrative theoretical background

#### **2.1. Distributed System**

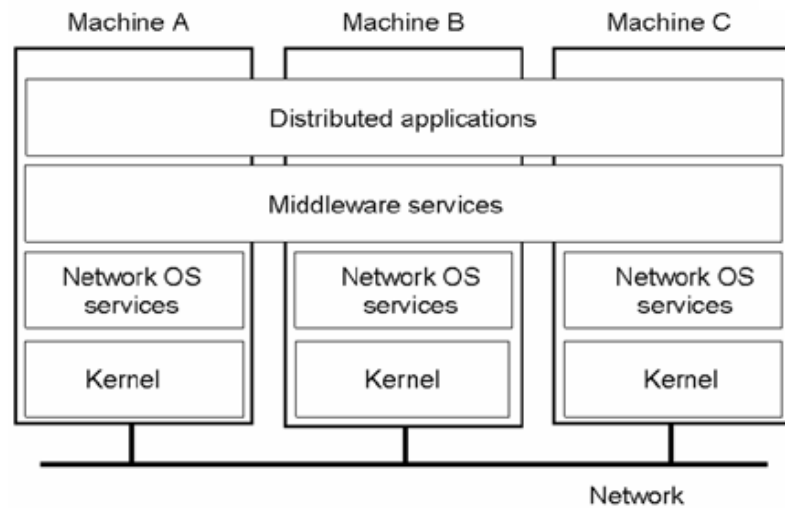
##### **2.1.1. Definition of A Distributed System**

A distributed system is a collection of independent computers that appears to its users as a single coherent system. A distributed system organized as middleware. The middleware layer extends over multiple machines and offers each application the same interface as shown if figure 2.1 [8].

##### **2.1.2. Distributed System Goals [6]**

- **Making Resources Accessible:** The main goal of a distributed system is to make it easy for the users (and applications) to access remote resources, and to share them in a controlled and efficient way.
- **Distribution Transparency:** An important goal of a distributed system is to hide the fact that its processes and resources are physically distributed across multiple computers.
- **Openness:** An open distributed system is a system that offers services according to standard rules that describe the syntax and semantics of those services.
- **Scalability** of a system can be measured along at least three different dimensions:
  - A system can be scalable with respect to its size, meaning that we can easily add more users and resources to the system.

- A geographically scalable system is one in which the users and resources may lie far apart.
- A system can be administratively scalable.



**Figure 2.1** Distributed System Model [30]

### 2.1.3. Types of Distributed Systems

There are three types of Distributed Systems.

- **Distributed Computing Systems:** an important class of distributed systems is the one used for high-performance computing tasks.
- **Distributed Information Systems:** is found in organizations that were confronted with a wealth of networked applications.
- **Distributed Pervasive Systems:** Pervasive systems are created by introducing wireless communication into distributed multimedia systems. These systems facilitate mobile data access to applications such as health care, tourism and emergency [8].

#### 2.1.4. Communication of Distributed System

Communication in distributed systems is always based on low-level message passing as offered by the underlying network [8]. Under this concept there are four types of communication that take place over distributed systems such types are variant in communication over distributed system.

1. **Remote Procedure Call:** When a process on machine A calls a procedure on machine B, the calling process on A is suspended, and execution of the called procedure takes place on B. Information can be transported from the caller to the call in the parameters and can come back in the procedure result. No message passing at all is visible to the programmer.
2. **Message-Oriented Communication:** when it cannot be assumed that the receiving side is executing at the time a request is issued, alternative communication services are needed. Likewise, the inherent synchronous nature of RPCs, by which a client is blocked until its request has been processed.
3. **Stream-Oriented Communication:** this type of communication is that it does not matter at what particular point in time communication takes place.
4. **Multicast Communication:** sending data to multiple receivers called also multicast communication.

#### 2.1.5. Distributed System Security

Distributed systems apply all security policies and mechanisms that provide authentication, confidentiality, availability, integrity and control access. The secure channels should provide authentication, message integrity and confidentiality, and secure group communication. Firewalls and secure mobile code are used for secure access control.

Security can have varying levels of difficulty for implementation. One factor in determining the difficulty is the number and distribution of the systems. When only individual systems need to be protected, such as one computer with all files residing locally and with no need to connect to any outside resources, security is not as complex as with distributed systems. With distributed systems architecture, there are different nodes and resources. One major issue with distributed systems is application security [24].

## **2.2. Mobile agent**

### **2.2.1. Definition of an agent system**

An agent system is a platform that can create, interpret, execute, transfer and terminate agents. Like an agent that is associated with an authority that identifies the person or organization for which the agent system acts. Where it is uniquely identified by its name and address. While host can contain one or more agent systems [9].

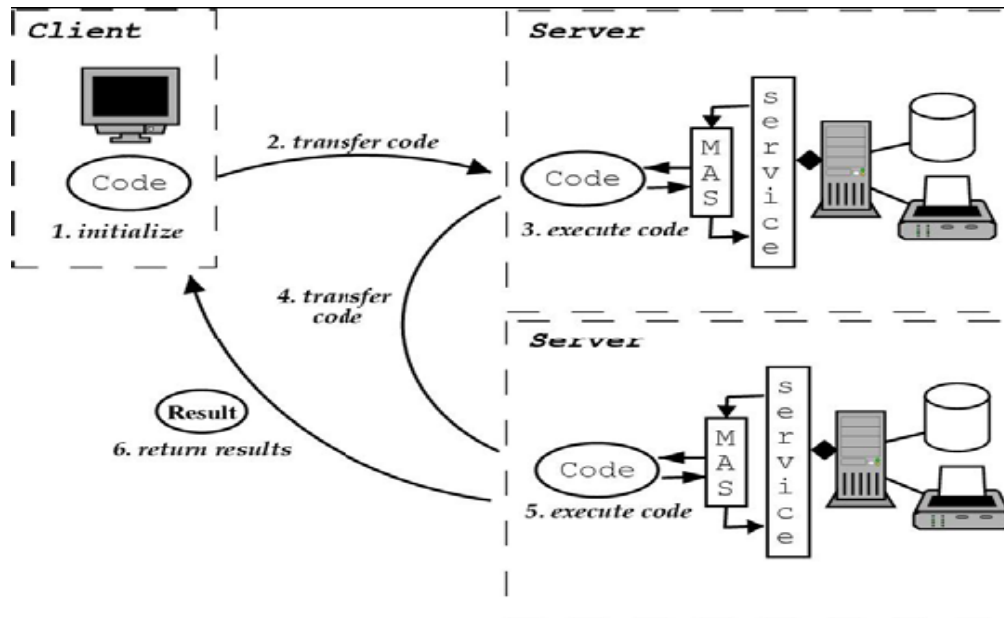
We can say for an agent as a computer program whose purpose is to help a user perform some task (or set of tasks). To do this, it contains persistent state and can communicate with its owner, other agents and the environment in general. Agents can do routine work for users or assist them with complicated tasks; they can also mediate between incompatible programs and thus generate new, modular and problem-oriented solutions, saving work [10].

### 2.2.2 Definition of Mobile agent

A mobile agent is a particular class of agent with the ability during execution to migrate from one host to another where it can resume its execution. A mobile agent is not bound to the system where it begins execution. It has the unique ability to transport itself from one system in a network to another. Agent system that contains an object which the agent want to interacts with. Moreover, the agent may utilize the object services of the destination agent system [9].

**Mobile Agent technology consists of mainly two components:** the mobile agent system and the mobile agents. The mobile agent system hosts the agents and provides an environment where they are executed after their arrival. The agents move from agent system to agent system on their own volition and work disconnected from their original home computer [9].

A common mobile agent interaction is depicted in Figure 2.2; it shows the behavior of a mobile agent at the network level. In step one the mobile agent is initialized on the client. Then the mobile agent moves via the network to the next mobile agent system.



**Figure 2.2** The Mobile Agent Paradigm [9]

There it will be executed and uses the local environment. While being executed at the server the agent can stop its execution and move to another server. Again, the agent's code is transmitted via the network to the new server, where the mobile agent will be executed. After it has finished its computation the mobile agent sends the results back to the user [9].

### 2.2.3 Mobile agent Components:

A mobile agent contains the following 3 components:

1. **Code** - the program (in a suitable language) that defines the agent's behavior.
2. **State** - the agent's internal variables, which enable it to resume its activities after moving to another host.

3. **Attributes** - information describing the agent, its origin and owner, its movement history, resource requirements, authentication keys . Part of this may be accessible to the agent itself, but the agent must not be able to modify the attributes [10].

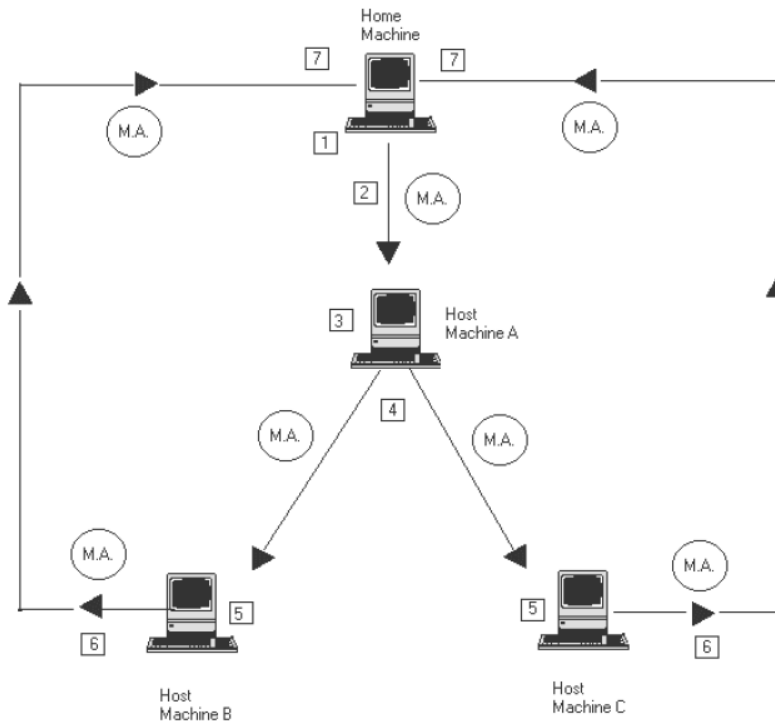
Mobility increases the functionality of the mobile agent and allows the mobile agent to perform tasks beyond the scope of static agents.

#### **2.2.4 Mobile agent life cycle:**

The figure 2.3 describe the life cycle of the mobile agent as:

1. **Creation:** A brand new agent is born and its state is initialized.
2. **Dispatch:** An agent travels to a new host.
3. **Cloning:** A twin agent is born and the current state of the original is duplicated in the clone.
4. **Deactivation:** An agent is put to sleep and its state is stored on a disk of the host.
5. **Activation:** A deactivated agent is brought back to life and its state is restored from disk.
6. **Retraction:** An agent is brought back from a remote host along with its state to the home machine.
7. **Disposal:** An agent is terminated and its state is lost forever.





**Figure 2.3** Mobile Agent Life Cycle [31]

### 2.2.5 Mobile agent security requirements:

1. **Confidentiality:** - sensitive data must be secure. So any private data stored on a platform or carried by an agent must remain confidential. Agent frameworks must be able to ensure that their intra- and inter-platform communications remain confidential.
2. **Integrity:** - altering data must be detected. So the agent platform must protect agents from unauthorized modification of their code, state, and data and ensure that only authorized agents or processes carry out any modification of shared data.

The agent itself cannot prevent a malicious agent platform from tampering with its code, state, or data, but the agent can take measures to detect this tampering.

3. **Authentication:** - an agent must authenticate it self to the host, and an agent server must authenticate it self to the agent.

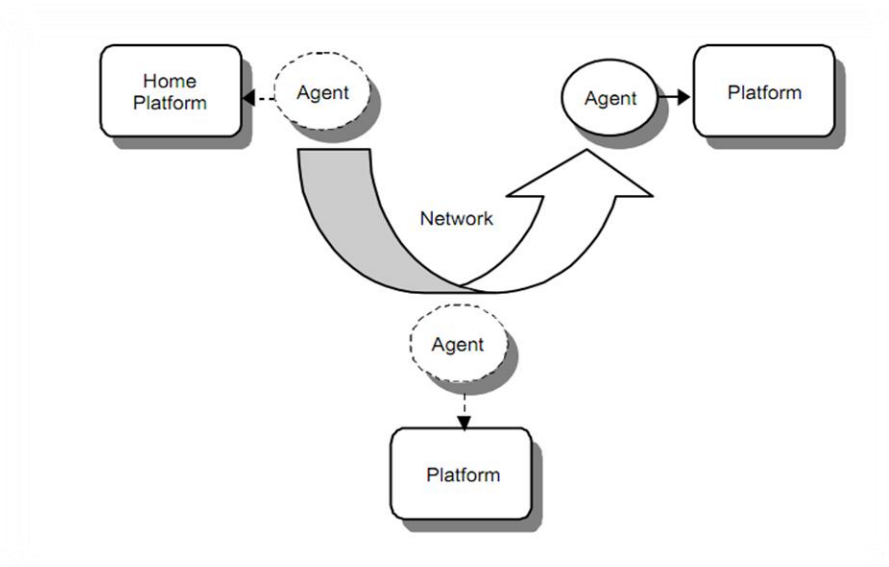
4. **Authorization:** - host enforces strict access control to its resources.
5. **Auditing:-** keeping track of the system, if an agent misbehaves, this should be logged.
6. **Availability:** The agent platform must be able to ensure the availability of both data and services to local and remote agents. The agent platform must be able to provide controlled concurrency, support for simultaneous access, deadlock management, and exclusive access as required.
7. **Accountability:** Each process, human user, or agent on a given platform must be held accountable for their actions. In order to be held accountable each process, human user, or agent must be uniquely identified, authenticated, and audited [11.]

### 2.3 Security Issues in Mobile Agent

Number of issues threaten the mobile agent work. This section will declare these threats and the type of danger and damage they cause.

The National Institute of Standards and Technology (NIST) sp800-19 publication was addressing these threats in detailed manner so we depend on this publication for defining and refutation of the mobile agent security threats.

Security threats can be classified into three classes, disclosure of information, denial of service, and corruption of information [11]. We will depend on the agent system component to identify the source and the destination of the attack. Using a very simple agent system composed of agent and agent platform as shown in figure 2.4.



**Figure 2.4** Simple mobile agent s system [11]

Code and status information comprise the agent, mobility allow agent to move among agent platform, while the platform provide the computational environment in which an agent operates. The platform from which an agent originates is referred to as the home platform, and normally is the most trusted environment for an agent [11].

**Four threat categories are identified:**

- Threats stemming from an agent attacking an agent platform.
- An agent platform attacking an agent.
- An agent attacking another agent on the agent platform.
- Other entities attacking the agent system [11].

Before starting in details of each one of them we shall explore some common security terminologies:

- **Denial of Service (DoS):** is an action that prevents or impairs the authorized use of networks, systems or applications by exhibiting resources such as central processing unit (CPU), memory, bandwidth, and disk space [13].
- **Masquerading:** masquerade is a type of attack where the attacker pretends to be an authorized user of a system in order to gain access to it or to gain greater privileges than they are authorized for. A masquerade may be attempted through the use of stolen logon IDs and passwords, through finding security gaps in programs, or through bypassing the authentication mechanism [14].
- **Unauthorized Access:** is when a person who does not have permission to connect to or use a system gains entry in a manner unintended by the system owner [16].
- **Repudiation:** when user sending data or a user denies receiving or possessing the data [15].
- **Eavesdropping:** The act of secretly listening to the private conversation of others without their consent. As a real word example man-in-the-middle attack (MITM) is an eavesdropping attack [17][18].
- **Alteration (Modification of Message):** Means that some portion of legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect [19].
- **Copy and Replay:** involves passive capture of data unit and its subsequent retransmission to produce an unauthorized effect [19].

After this tour, now go to start the main event of this section detailing the mobile agent threats.

### **3.2.1 Agent –to- platform threats**

The agent-to-platform category represents the set of threats in which agents exploit security weaknesses of an agent platform or launch attacks against an agent platform. This set of threats includes masquerading, denial of service and unauthorized access.

#### **3.2.1.1 Masquerading**

When agent claims the identity of another agent and does unauthorized effects to agent platform. and sticks the blame to the real one.

#### **3.2.1.2 Denial of Service**

This happen by consuming an excessive amount of the agent platform's computing resources. This attack may be caused by script designed for attack purpose or by programming errors.

#### **3.2.1.3 Unauthorized Access**

Since every system has access control privileges the agent platform must have ones. When these access control privileges are broken then this considered as unauthorized access.

### **3.2.2 Agent –to- Agent threats**

The agent-to-agent category represents the set of threats in which agents exploit security weaknesses of other agents or launch attacks against other agents. This set of threats includes masquerading, unauthorized access, denial of service and repudiation.

#### **3.2.2.1 Masquerading**

Since communication can be established between agents then the possibility of this type of threats is presented. In some way masquerading in this part became like spoofing.

#### **3.2.2.2 Denial of Service**

As agents can gain a DoS attack against agent's platform they also can attack other agents by this type of attacks. Sending a huge amount of messages to the targeted agent will obviously cause a denial of service.

#### **3.2.2.3 Repudiation**

This happens when an agent is repudiating doing a transaction or communication that was really done.

#### **3.2.2.4 Unauthorized Access**

If the agent platform has weak or no control mechanisms in place, an agent can directly interfere with another agent by invoking its public methods (e.g. attempt buffer overflow, reset to initial state, etc.), or by accessing and modifying the agent's data or code. Modification of an agent's code is a particularly insidious form of attack, since it can radically change the agent's behavior (e.g., turning a trusted agent into malicious one). An agent may also gain information about other agents' activities by using platform services to eavesdrop on their communications.

#### **3.2.3 Platform –to- Agent threat**

The platform-to-agent category represents the set of threats in which platforms compromise the security of agents. This set of threats includes masquerading, denial of service, eavesdropping, and alteration.

##### **3.2.3.1 Masquerade**

When agent platform is masquerading as another platform to gain unauthorized effects, collecting data and cause harm to the system.

##### **3.2.3.2 Denial of Service**

When dealing with malicious platforms they can ignore a request of agent and delay this request causing a lose by making the agent wait for the acknowledgment.

### **3.2.3.3 Eavesdropping**

The agent platform can monitor communication and every instruction executed by the agent then the, all public and unencrypted data are also available to these platforms and this is cause a source of danger and real threat.

### **3.2.3.4 Alteration**

When the host platform trying of alternating the mobile agent code or status. This will cause a great harm to the system either by joining a malicious code to the agent's code or replace the agent status information with fake ones.

### **3.2.4 Other –to- agent platform threats**

The other-to-agent platform category represents the set of threats in which external entities, including agents and agent platforms, threaten the security of an agent platform. This set of threats includes masquerading, denial of service, unauthorized access, and copy and replay.

#### **3.2.4.1 Masquerading**

Remote users, processes, and agents may request resources for which they are not authorized. Remote access to the platform and the host machine itself must be carefully protected, since conventional attack scripts freely available on the Internet can be used to subvert the operating system and directly gain control of all resources. Remote administration of the platform's attributes or security policy may be desirable for an administrator that is responsible for several distributed



platforms, but allowing remote administration may make the system administrator's account or session the target of an attack.

#### **3.2.4.2 Unauthorized Access**

Remote users, processes, and agents may request resources for which they are not authorized. Remote access to the platform and the host machine itself must be carefully protected, since conventional attack scripts freely available on the Internet can be used to subvert the operating system and directly gain control of all resources. Remote administration of the platform's attributes or security policy may be desirable for an administrator that is responsible for several distributed platforms, but allowing remote administration may make the system administrator's account or session the target of an attack.

#### **3.2.4.3 Denial of service**

Agent platform services can be accessed both remotely and locally. The agent services offered by the platform and inter-platform communications can be disrupted by common denial of service attacks. Agent platforms are also susceptible to all the conventional denial of service attacks aimed at the underlying operating system or communication protocols.

#### **3.2.4.4 Copy and Reply**

Every time a mobile agent moves from one platform to another it increases its exposure to security threats. A party that intercepts an agent, or agent message, in transit can attempt to copy the agent, or agent message, and clone or retransmit it. For example, the interceptor can capture an agent's "buy order" and replay it several times, having the agent buy more than the original agent had intended. The interceptor may copy and replay an agent message or a complete agent.

## **2.4 Software Description**

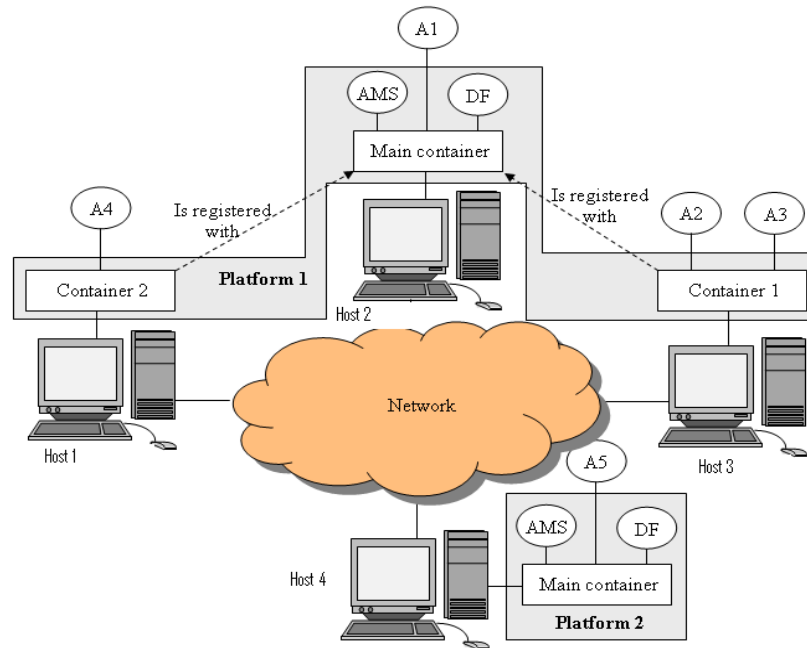
### **2.4.1. The JADE Platform**

Java Agent Development Framework (JADE) is a software platform that provides basic middleware-layer functionalities which are independent of the specific applications that exploit the software agent abstraction.

#### **2.4.1.1. JADE Architecture**

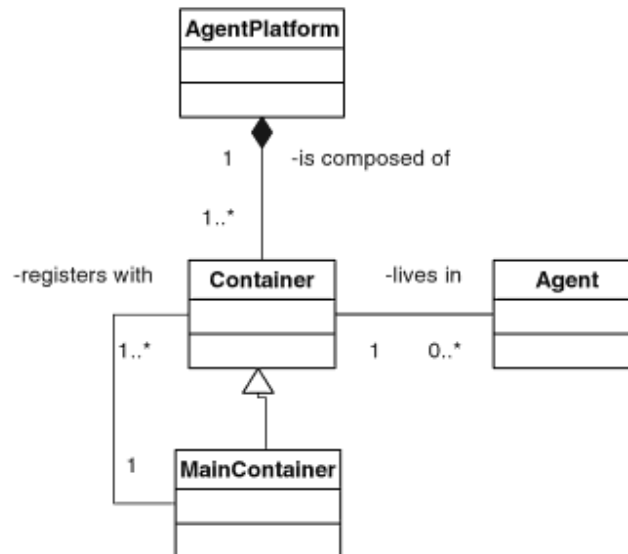
Figure 2.5 represents the main JADE architectural elements. An application based on JADE is made of a set of components called Agents each one having a unique name. Agents execute tasks and interact by exchanging messages. Agents live on top of a Platform that provides them with basic services such as message delivery. A platform is composed of one or more Containers. Containers can be executed on different hosts thus achieving a distributed platform. Each container can contain zero or more agents.

A special container called Main Container exists in the platform. The main container is itself a container and can therefore contain agents, but differs from other containers as: It must be the first container to start in the platform and all other containers register to it at bootstrap time. It includes two special agents: the AMS that represents the authority in the platform and is the only agent able to perform platform management actions such as starting and killing agents or shutting down the whole platform (normal agents can request such actions to the AMS). And The DF (Directory Facilitator) that provides a Yellow Pages service by means of which an agent can find other agents providing the services he requires in order to achieve his goals.



**Figure 2.5.** Main architectural elements of a JADE platform [20]

The UML diagram in figure 2.6 schematizes the relationships between the main architectural elements of JADE.



**Figure 2.6.** Relationship between the main architectural elements[21]

#### **2.4.1.2 Foundation of intelligent physical agents (FIPA)**

FIPA was originally formed as a Swiss based organization in 1996 to produce software standards specifications for heterogeneous and interacting agents and agent based systems. Since its foundations, FIPA has played a crucial role in the development of agents standards and has promoted a number of initiatives and events that contributed to the development and uptake of agent technology. Furthermore, many of the ideas originated and developed in FIPA are now coming into sharp focus in new generations of Web/Internet technology and related specifications.

#### **2.4.2. Java programming language**

Java is a programming language originally developed by James Gosling at Sun Microsystems (which is now a subsidiary of Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to bytecode (class file) that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere". Java is currently one of the most popular programming languages in use, and is widely used from application software to web applications.

#### **2.4.3 Java platform, standard edition (Java SE)**

There are two principal products in the java SE platform family: Java SE Runtime Environment (JRE) and Java development kit (JDK).

- **The Java Runtime Environment (JRE)**, provides the libraries, the Java Virtual Machine, and other components to run applets and applications written in the Java programming language [22].

- **Java development kit (JDK)**

The JDK is a superset of the JRE, and contains everything that is in the JRE, plus tools such as the compilers and debuggers necessary for developing applets and applications [23].

### **2.4.3. NetBeans IDE**

NetBeans refers to both a platform framework for Java desktop applications, and an integrated development environment (IDE) for developing with Java, JavaScript, PHP, Python, Ruby, Groovy, C, C++, Scala, Clojure, and others. The NetBeans IDE is written in Java and can run anywhere a JVM is installed, including Windows, Mac OS, Linux, and Solaris. A JDK is required for Java development functionality, but is not required for development in other programming languages. The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans platform (including the NetBeans IDE) can be extended by third party developers [25].

# **Chapter Three**

## **Conceptual Design**

**3.1 Detailed Project Objectives**

**3.2 Security Solutions**

**3.3 Testing Application**

## **Chapter Three**

### **Conceptual Design**

This chapter contains the detailed project objectives the explanation and conceptual design of security solutions and testing application.

#### **3.1 detailed project objectives**

- Making survey about the mobile agent common security (vulnerabilities) and threats.

Different types of security problems face the mobile agent based application as explained in section 2.3. This project addresses these problems in order to find a solution for them.

- Search for acceptable solutions for these security threats.

Security solutions always adds some overload to the system .This project will try to balance between the load that security solutions add to the system and their efficiency.

- Implement these solutions with Java programming language to be suitable to use over JADE mobile agent development environment.

In this project the JADE (JAVA Agent Development Framework) was chosen as mobile agent development environment. As described in section 2.4 JADE was built with java programming language. So, the implementation of the solutions will be written in java.

- Design simple mobile agent based application.

File Name Searcher (FNS) is a simple mobile agent based application that takes file name from user as input, search for that file over network and returns the file location and other information as output.

- Implement the FNS as JADE multi agent application.

To test the proposed security solutions areal testing should be done. So; the FNS was chosen for this purpose.

## **3.2 The Security Solution**

In this section we discuss the security solution that this project proposes for securing JADE over intra connected platform.

### **3.2.1 System Definition**

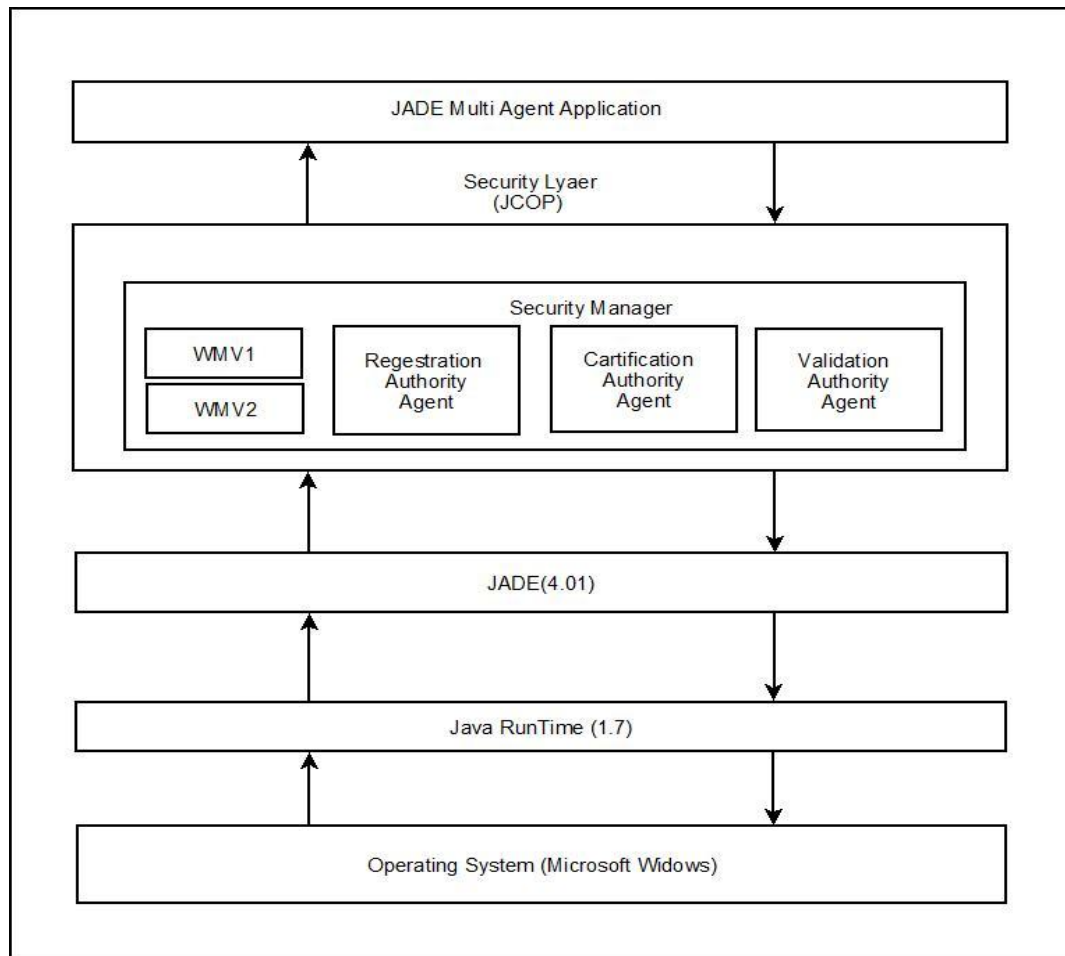
JCOP is a framework that provides security libraries and secure infrastructure for JADE over intra connected platform. This framework contains a set of agents that work to provide the security for the platform and a set of classes packaged in a library to enable programmer to run his/her agents over JADE.

### **3.2.2 System Block diagram**

There are three main blocks in System:

- Security Manager (SM)
- JADE Multi Agent Application.





**Figure 3.1:** System Block Diagram

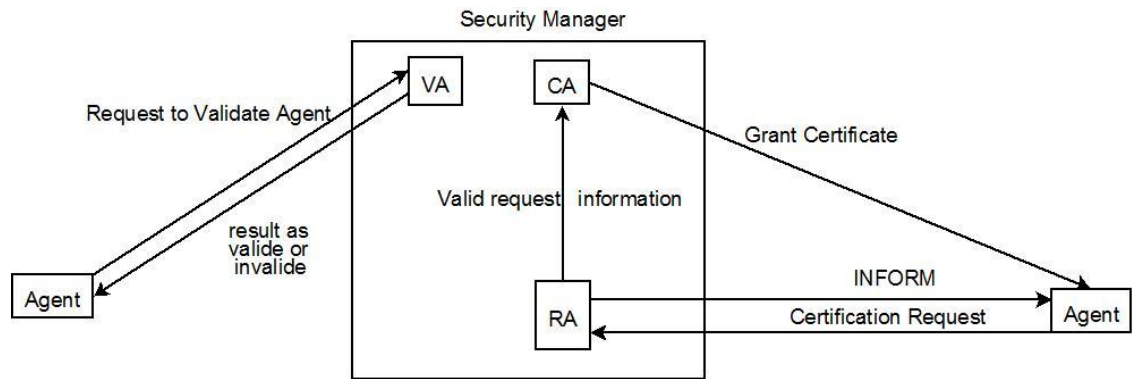
### 3.2.2.1 JADE Multi Agent Application (Agent)

Agent can be either stationary or mobile Agent that migrates from container to another. This Agent is authenticated and validated from the SM that will be declared next. When agent created it receives inform message from the SM that it must be certified (Authenticated) to continue working in the platform. In response to this the agent sends a message that describes its behavior and asks the SM for certificate.

### 3.2.2.3 Security Manager (SM)

This Module contains three Stationary Agents that resides in the main container. These agents cooperate together to provide security to the system. These agents are:

- Registration Authority (RA).
- Certification Authority (CA).
- Validation Authority (VA).
- WMV1
- WMV2



**Figure 3.2:** Security Manager Block diagram

#### 3.2.2.3.1 Registration Authority (RA)

When any agent is created it receives inform message from the RA if the agent dose not respond to this message the RA kills this agent. If the agent responds, then its response sent back as message that contains information that describes the agent work. These information are processed by a special processor (program) that parse the message text searching for specific words that makes the agent illegal. These words are stored in black list predefined in programming time and have the ability to gust other words added by the administrator. If the Agent classified as illegal then RA kills this agent. If agent classified as legal then the RA send a message that contains the validated information about the agent to CA to get this agent Certified

#### **3.2.2.3.2 Certification Authority (CA)**

When this Agent receives a message from the RA that it generates a certificate that contains the information in the message signed by the CA private key along with the CA public key. The Certificate is stored in a folder called Certificate Store as files.

#### **3.2.2.3.3 Validation Authority (VA)**

This agent needed when agent need to validate another agent to exchange information with. This agent connects to the Certificate Store and checks if it has the certain certificate and validate it. If Certificate found then VA validate it.

#### **3.2.2.3.4 Watchman Agents**

Both agents Watchman version 1(WMV1), and Watchman version 2(WMV2) are both introspector agents, where WMV1 introspect the agent creation agents and WMV2 introspect the agents movements.

### **3.2.3 Problem Solving**

The security scheme in this project focuses on two main security roles to secure the System authentication and access control.

- Authentication: authenticate the agent to remote containers and to the platform
- Access Control: the SM agents having the highest privileged roles( full control over the System) and other agents having lower level of access control

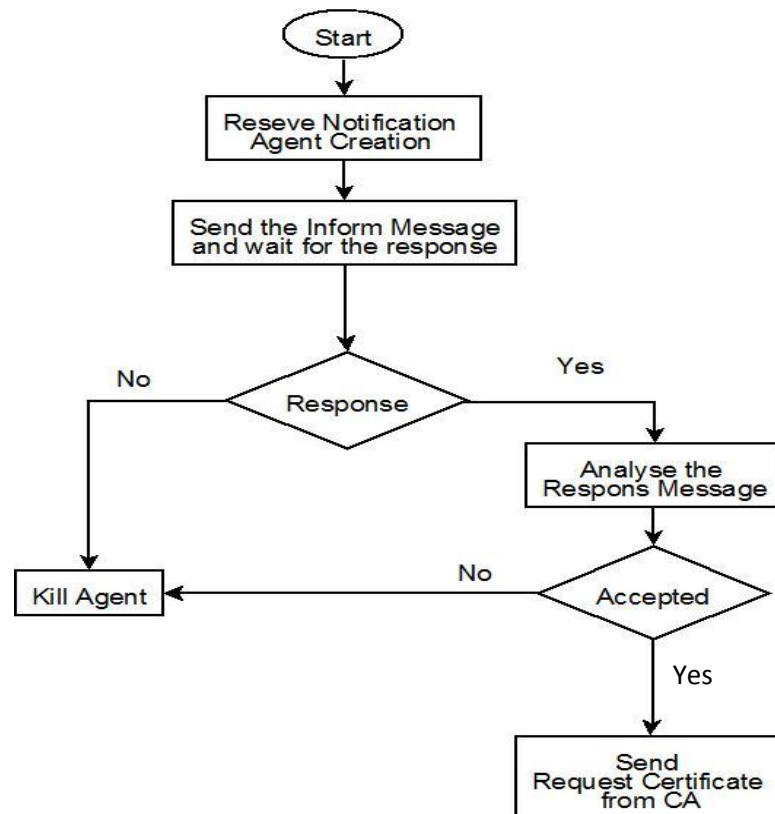
How Authentication and Access Control provide security to the System?

- Only authenticated agent is allowed to live and to operate in agent platform.

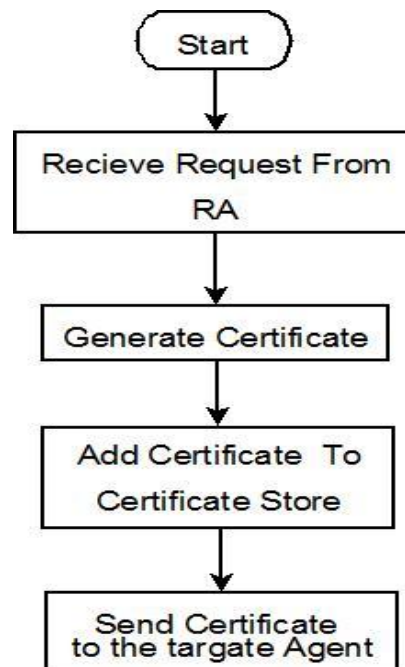
- When agents are created they must apply certain rules. If they are not then they are killed by SM.
- To move or to do administrative acts the security manager acts as observer.

### 3.2.4 System Modeling

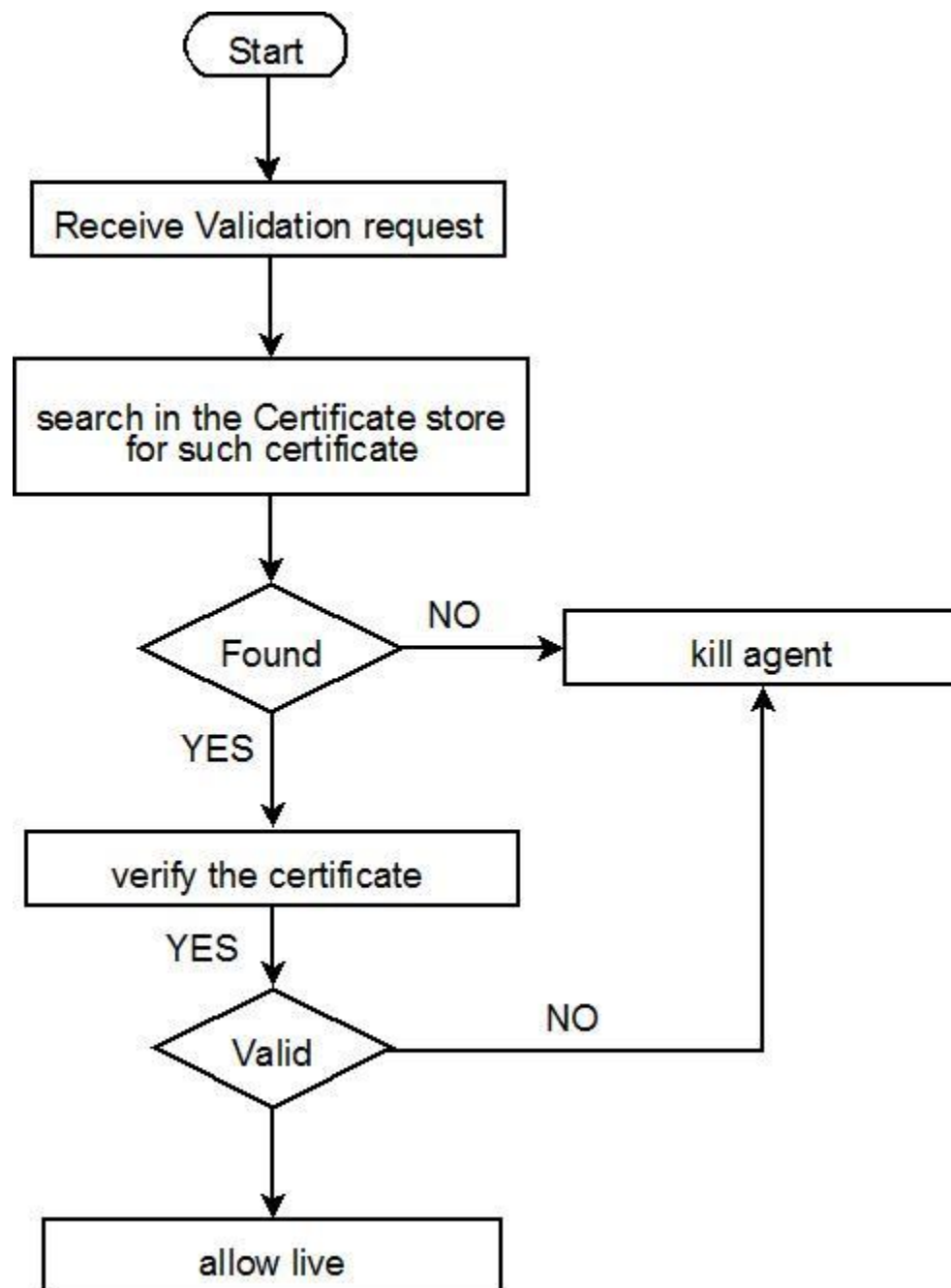
This section describes the system behavior using flowcharts.



**Figure 3.3:** Flowchart Describes RA Agent Behavior



**Figure 3.4** Flowchart Describes The CA Agent Behavior



**Figure 3.5** Flowchart Describes The VA Agent Behavior

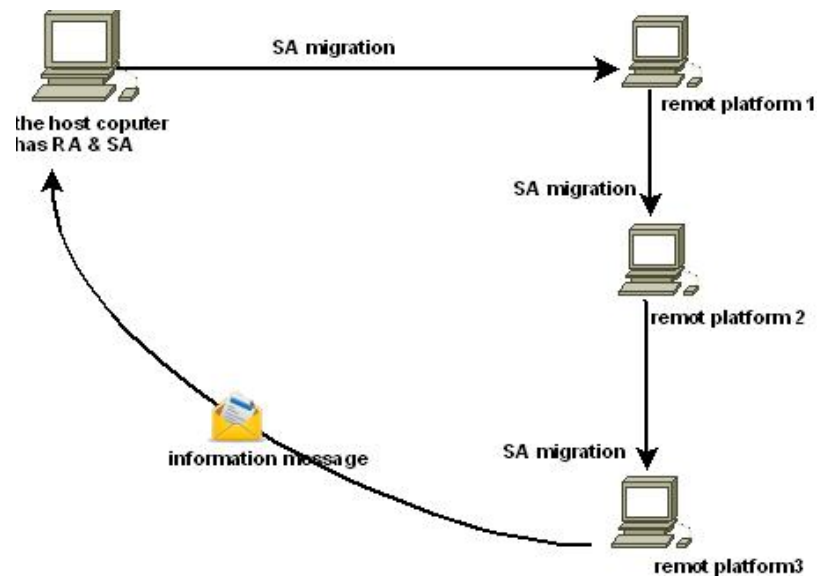
### 3.3 Testing Application

This section will contain the conceptual design issues, architectural design, and system modeling for the testing application.

#### 3.3.1 System Definition

FNS (File Name Searcher) takes the advantage of mobile agents to migrate to platforms and get the file name. It consist two agents, the first one is a mobile agent which name is Searcher agent (SA) that search about the file name. And the second is stationary agent which name is reference (RFA) which stays in the host platform. When RA get name of file then it create SA that carry the file name, it migrate from one hop to another until it get the specific information then it send those information to the RA.

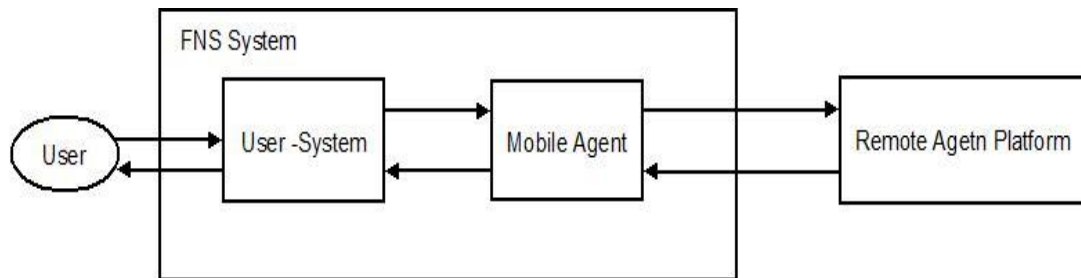
The project team selects this application because it is highly use the mobility service. That the project aims to secure.



**Figure 3.6** General idea of the FNS

### 3.3.2 System Block Diagram

As Shown in Figure 3.10 below describes the File Name Searcher (FNS) system application



**Figure 3.7:** System Block Diagram

#### 3.3.2.1 User System (interaction system)

Contains GUI that provides user with different options:

- Allows user to submit the target file name.
- Option to start the search process.
- Option to stop the search process if it is needed.

This subsystem contains a stationary agent called Reference Agent (RFA). Its job is to get file name and organize work of mobile agent.

The RA interacts with mobile agent with deferent ways such as:

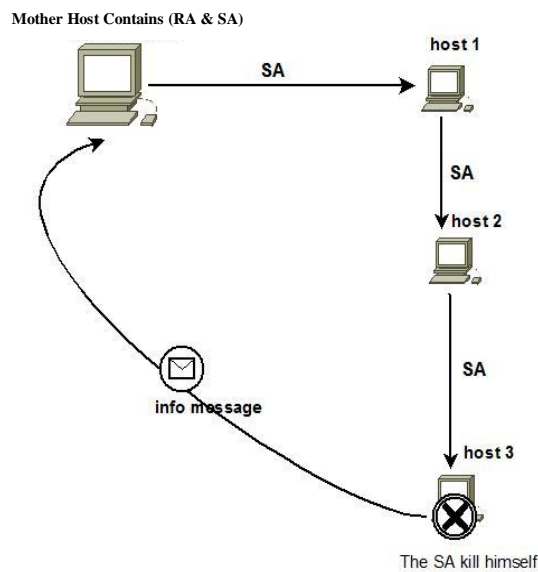
- Provides the mobile agent with migration plan.
- Provides mobile agent with target file name.
- Activates (create) and deactivate (kill) mobile agents



### 3.3.2.1 Mobile Agent

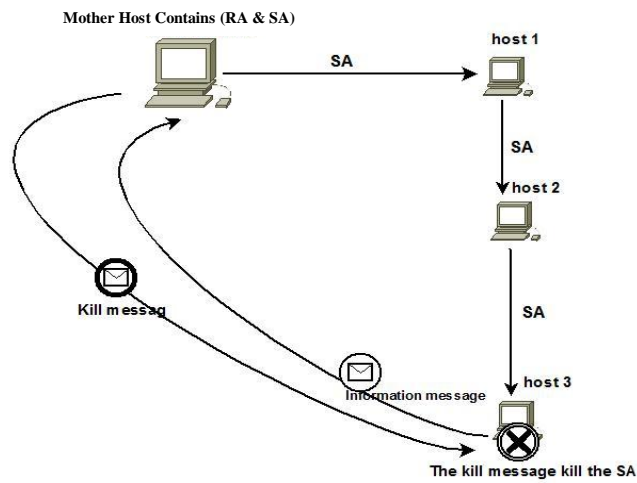
This system contains only one mobile agent called Searcher Agent (SA) which is created by a Stationary agent (RFA) in the main host. This mobile agent searches file name in another hosts and when it finds the file name, then three strategies will applied for :

- 1- The mobile agent sends information message as a file to the mother host and then kills itself. See Figure 3.8 bellow.



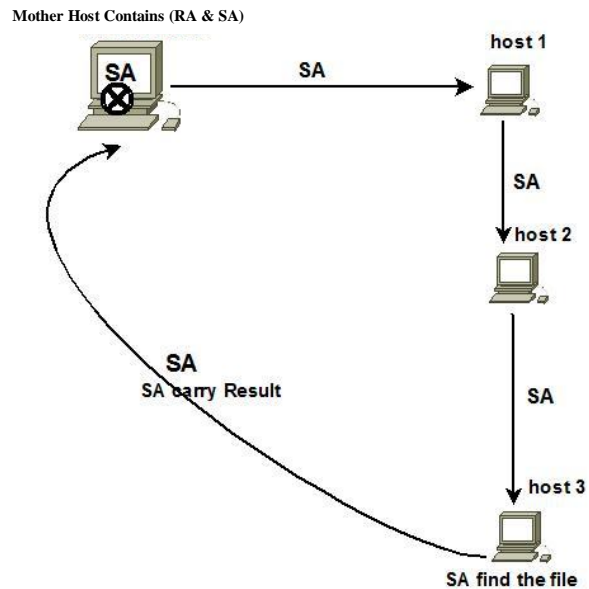
**Figure 3.8:** First Strategy of Mobile Agent in FNS

- 2- The mobile agent finds the file and sends the message then, the stationary agent in the main host sends kill message to him to be killed. As shown in figure 3.9 bellow.



**Figure 3.9:** Second Strategy of Mobile Agent in FNS

- 3- The mobile agent finds the file in the host and then sends it to the mother host, and then it kills the SA. As shown in figure 3.10 bellow.



**Figure 3.10:**Third Strategy of Mobile Agent in FNS

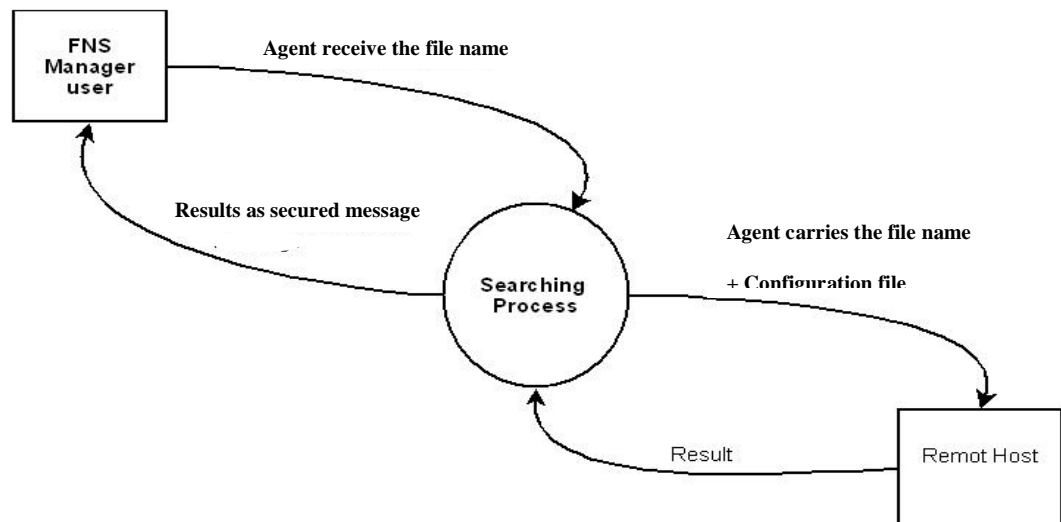
### 3.3.2.3 Remote Agent Platform

The Remote Agent platform receives the mobile agent in order to search for the file name. The mobile agent then uses a search algorithm to find the file name in the remote platform.

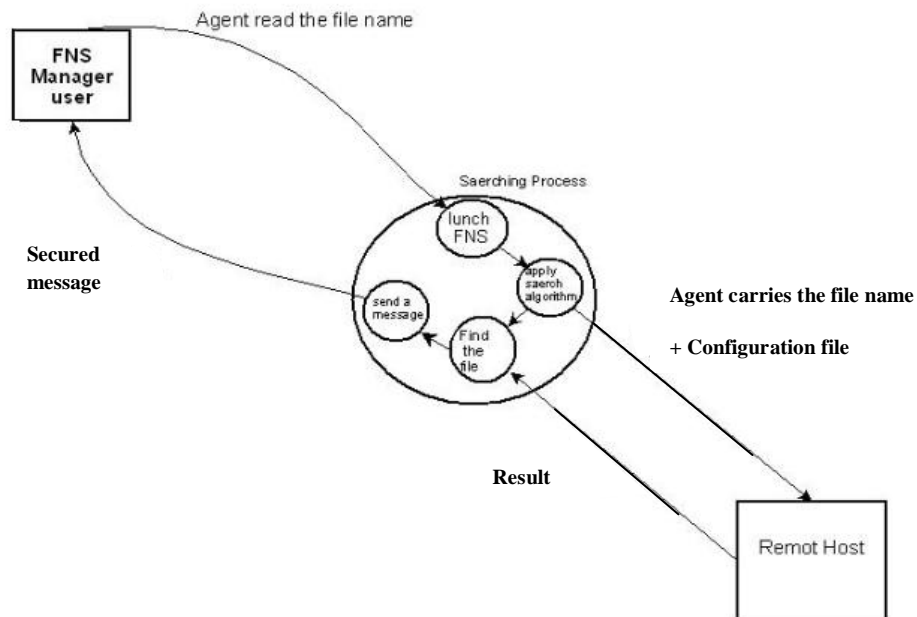
### 3.3.3 System Modeling

#### 3.3.3.1 Data flow diagram

Figur 3.11 Shows the dataflow diagram which describes the dataflow within the FNS application, the core processes which is shown in the circle is the searching process.



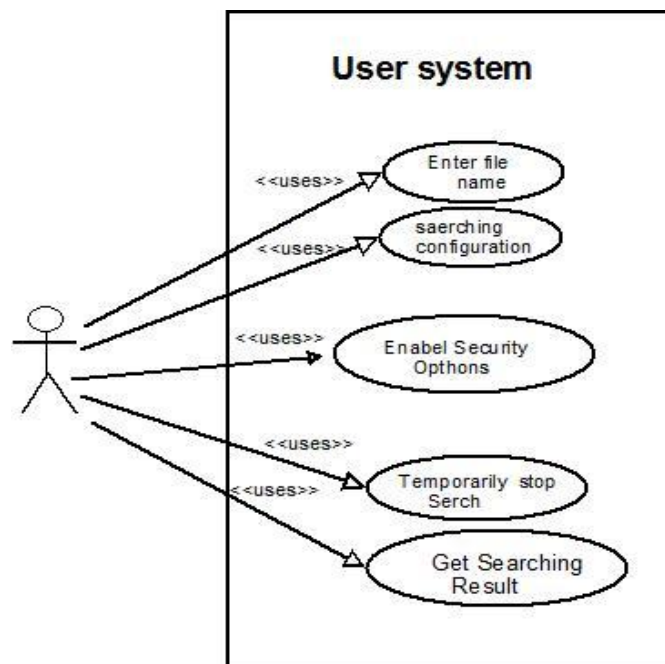
**Figure 3.11:** Show refined dataflow – level 0 which gives details for the searching process to explicitly describe each subprocess.



**Figure 3.12** Dataflow diagram – level 1

### 3.3.3.2 Use Cases

Figur 3.16 Use case diagram show the FNS application processes.



**Figure 3.13** Use case to FNS

# Chapter 4

## Detailed System Design

### **4.1 SM Architecture**

### **4.2 Detailed Specifications of SM components**

### **4.3 Agent Task and Interaction in JADE**

### **4.4 Detailed System Modeling**

### **4.5 Description of inter action classes**

### **4.6 FNS Architecture**

### **4.7 Limitations**

### **4.8 Assumptions**

### **4.9 Summary**

## **Chapter 4**

### **Detailed System Design**

In this chapter a detailed description of the design of different system components and modules, including system architecture, flow charts, GUI design, modules specifications, assumptions and limitations.

#### **4.1 Architecture of Security Manager(SM)**

The security manager SM consists of the following components:

- Agents
  - Registration Authority (RA).
  - Certification Authority (CA).
  - Validation Authority (VA).
  - WatchManV1 (WMV1).
  - WatchManV2 (WMV2).
  
- Programmer Interaction classes :
  - Certificate class
  - The Response Classes

The following figure shows the architecture of the system, including all components and their interactions

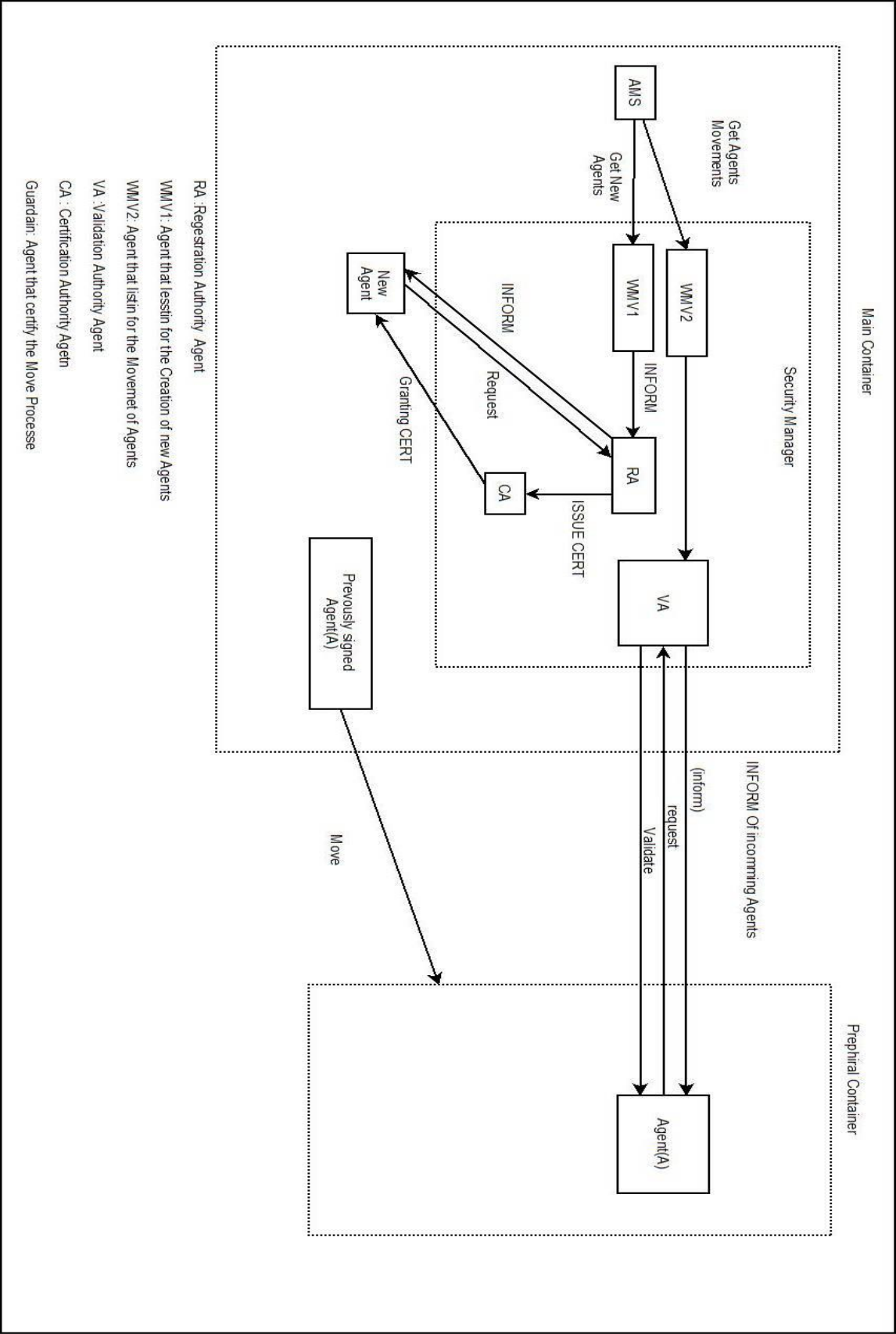


Figure 4.1 System Architecture

## **4.2 Detailed Specification of Security Manager Components**

### **4.2.1 Registration Authority (RA)**

- Only one RA agent is allowed running over the distributed platform.
- When new agents created in the platform RA informed by the WMV1.
- RA sends inform message to the new created agent to get certified.
- If the agent does not respond then RA kill this agent.
- If agent responds then RA classify the request valid then certify it or not then kill it.

### **4.2.2 Certification Authority (CA)**

- Only one CA agent allowed running over the distributed platform.
- At its start CA initialize pair public and private keys and store them in key store.
- When CA receives valid request message from the RA then it initiate a certificate folder carry the certificate id number in certificate store and initiate three files as follow
  1. Request Text: this file contains the request message.
  2. Signature: this file contains the signature of the message.
  3. Certificate (id): where id is the number of the certificate. This file considered as the official certificate. It contains the plan text and signature with some other data.
- Then sends a message that carries the certificate as object of type certificate.

For details about digital signature and digital certificates see [26,27,29]



### **4.2.3 Validation Authority**

- Only one RA agent allowed running over the distributed platform.
- This Agent validates any Agent that registered in the CA.
- When agent moves from container to another this container authenticate it.

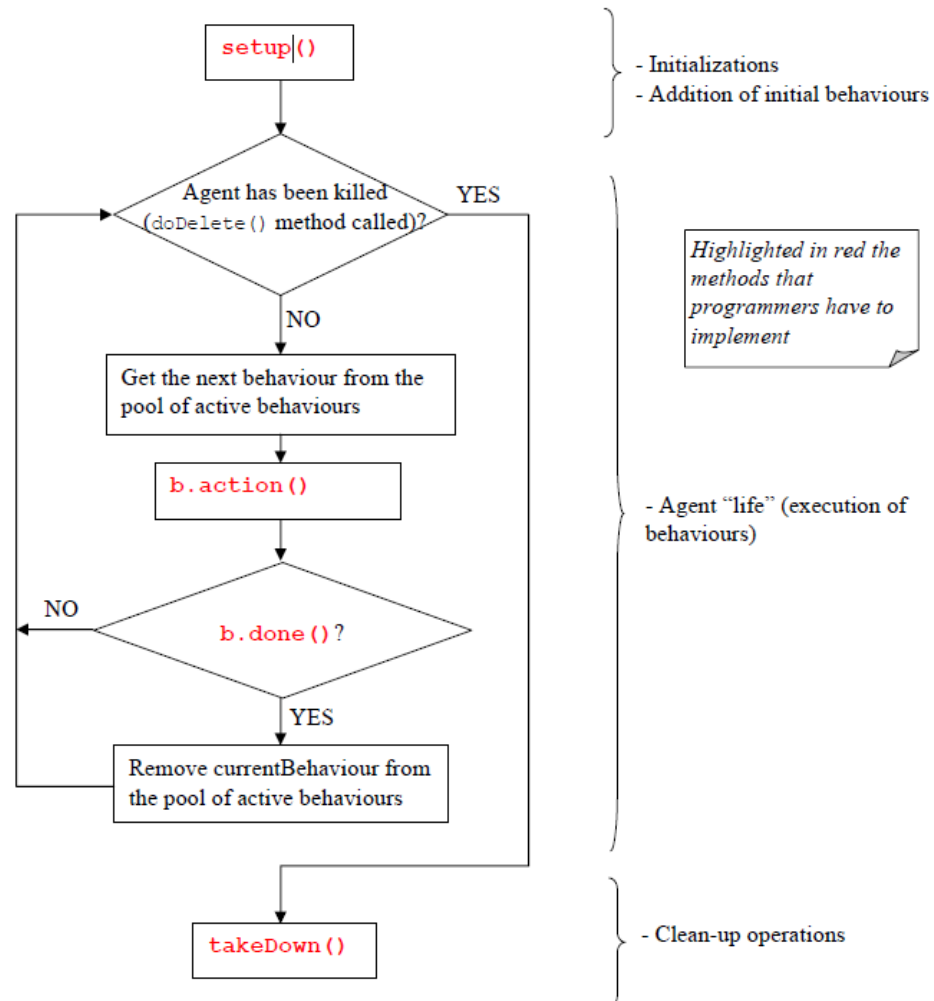
### **4.2.5 WatchManV1**

- Only one Agent of this type in the platform.
- This Agent inform the RA when new Agents Born.

### **4.2.6 WatchManV2**

- Only one Agent of this type in the platform.
- This Agent informs the VA that certain agent moves from container to another.

### 4.3 Agents Task Scheduling and Interactions in JADE



**Figure 4.2** Agent Thread Path of Execution. [29]

#### 4.3.1 Agent Task in JADE

The tasks that the agent will do are carried out within “behaviour”; each behavior is implemented as an object of a class extends `jad.core.behaviours`. To make the agent start a task an object that is implemented by the behavior; the behavior object should be added to the agent using the `addBehaviour` method. Behaviors should implement the abstract method `action()` that contains the operation the agent is to perform, and the `done()` method that return true if the behavior completed and false if it didn't.

#### 4.3.2 Primary types of behaviors in JADE, which are:

There are three primary types of behavior, which are:

1. **One-Shot behavior**, this type of behavior are designed to have one execution phase, that is their `action()` method executes only once the `JADE.core.behaviours.OneShotBehaviour` class already implements the `done()` method by returning true that is the behavior is done when the execution of `action()` method is done.

For example and implementation of a one-shot behavior class can be done using:

```
public class MyB extends OneShotBehaviour(){
    public void action(){
        //perform any operation
    }
}
```

2. **Cyclic Behavior**, this type of behaviors are designed never to complete, that is the action() method executes the same operation every time it's called, the done() method always returns false.

For example and implementation of a cyclic behavior class can be done using:

```
public class MyB extends CyclicBehaviour(){
    public void action(){
        //perform any operation
    }
}
```

3. **Generic behaviors**, see [21] “the JADE book”

#### 4.3.3 Scheduling operation

The package JADE.core.behaviours implements classes that do some operation one a selected point of time.

- WakerBehaviour has a method onWake() that executes after a given period of time.

**Example:**

```
public class MyAgent extends Agent{
    protected void setup(){
        System.out.println(" Adding Waker Behaviour");
        addBehaviour(new WakerBehaviour(this,10000){
            protected onWake(){
                //perform operation X
            }
        });
    }
}
```

```

    }
    });
    }

```

Operation X is performed after 10 seconds from adding the behavior

- TickerBehaviour has a method onTick() that executed repeatedly after a given period of time.

```

public class MyAgent extends Agent{
    protected void setup(){
        System.out.println ("Adding Ticker Behaviour");
        addBehaviour (new TickerBehaviour(this,10000){
            protected onTick(){
                //perform operation X
            }
        });
    }
}

```

Operation X is performed repeatedly every 10 seconds.

#### 4.3.4 Methods invoked during the agent life Cycle

There are different methods in JADE that are used to make the agent to different tasks at different point during its life cycle, these methods are:

- setup(): the setup() method is used to perform the agent initialization, it is the first method invoked when the agent is activated.
- afterMove(): this method is invoked when an agent moves to a container other than its current container.

- `afterClone()`: this method is invoked agent an agent is cloned; the method is called at the cloned agent, not the original one.
- `takedown()`: this method is invoked after the agent is terminated.

Other methods ate `beforeMove()`, `beforeClone()`.... etc. see [4].

#### **4.3.5 Interacting with AMS**

Since some agents in the system needs to do operations on the platform level, such as (WMV1) needs to be notified when new agents created in the platform, (WMV2) needs to be notified when agent moves from container to another , (RA) needs to kill illegal agents, guardian needs to kill unauthenticated agents. All these actions are done throw interaction with AMS (means agent cannot do these actions without AMS).

The AMS performs management duties related to the platform, such as killing and creating agents and containers ,it also stores (white pages) contains information about the platform and the agents, where an agent can request from AMS the currently available containers on the platforms, or the available agents on the container where the agent resides[4].

Any agent that wishes to perform platform management action must first request the AMS to perform them.

### **4.4 Detailed System Modeling**

In this section detailed functional description of the different system components, and the functions contained within each components, also a life cycle of the each agent is described using flowcharts.

#### 4.4.1 Security Manager Agents

##### 4.4 .1.1 Registration Authority (RA)

As defined before RA receives inform message from the WMV1 about new agents in the platform. And then act with them if they have a clear goal to live or not.

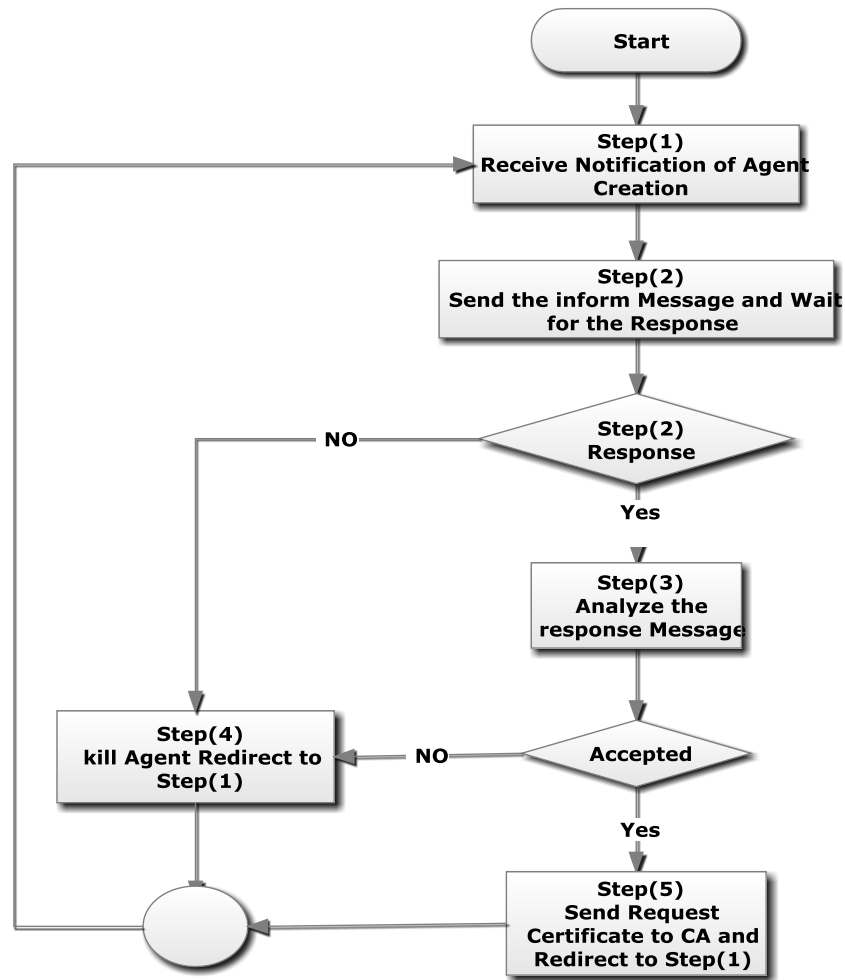
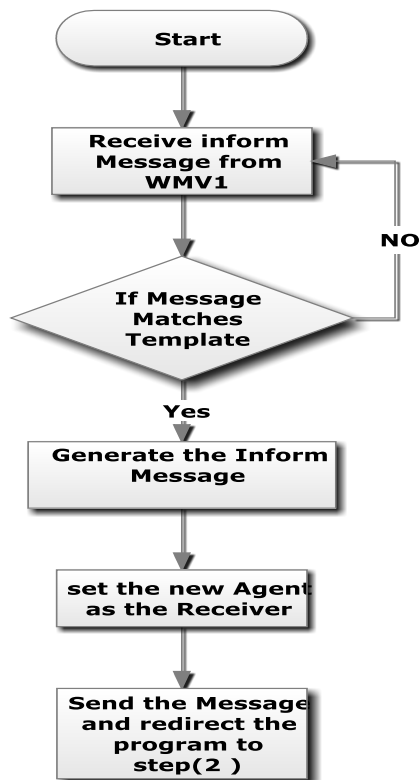


Figure 4.3 RA Global Behavior

<b>NAME</b>	Step(1)
<b>Input</b>	ACLMessage
<b>Output</b>	Redirection of program execution path to Step(2), ACLMessage
<b>Description</b>	this step represent the RA notification process from WMV1 and the RA inform for the newly created Agent



**Figure 4.4** RA Step (1)



<b>NAME:</b>	Step(2)
<b>Input</b>	Steps = response, ACLMessage
<b>Output</b>	Redirection of program execution path to Step(3) or to CA ACLMessage
<b>Description</b>	this step represent the RA response process where RA suppose to receives confirm message from the new agent if so then redirect program execution path to step (3) else redirect it to step(4)

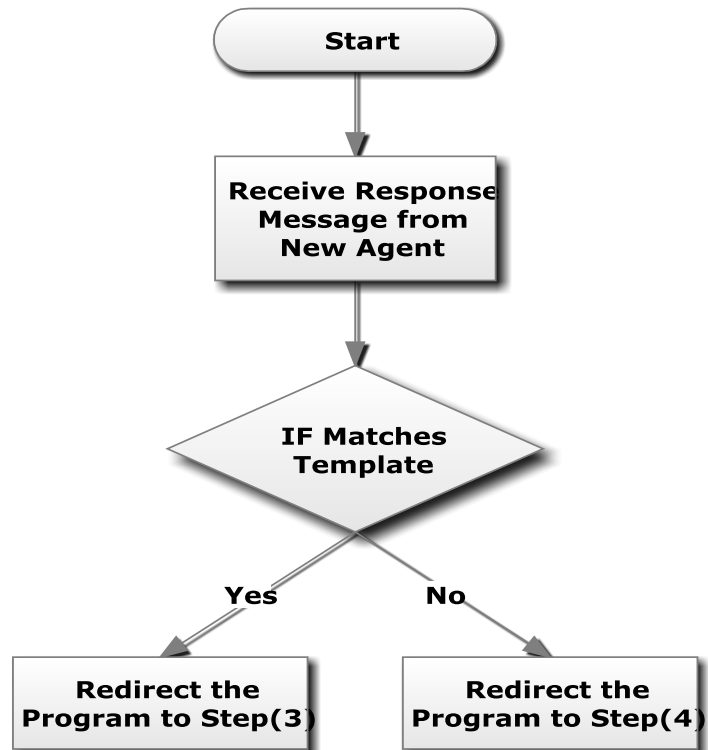
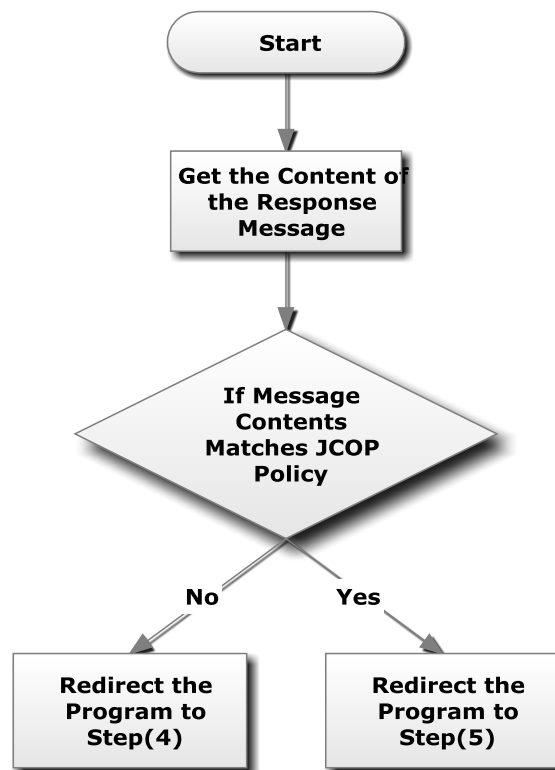


Figure 4.5 RA Step (2)

<b>NAME:</b>	Step(3)
<b>Input</b>	ACLMessage
<b>Output</b>	Boolean value true or false
<b>Description</b>	At this step the RA analyze the confirm message by comparing the message words with list of words known as the black list .if match occur then the agent considered as illegal and the program will redirected to step(4), if not then the program will be redirected to step(5)



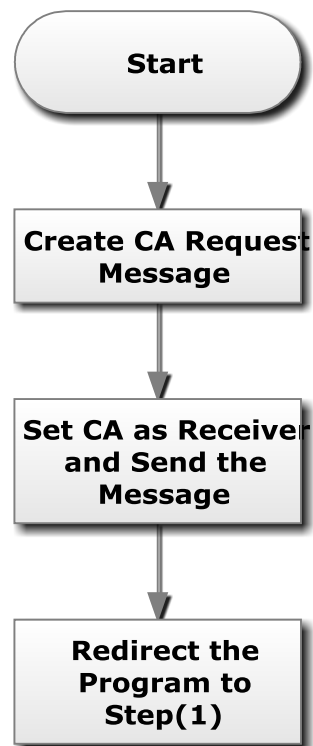
**Figure 4.6** RA Step (3)

NAME:	Step(4)
Input	Steps=goToHell (4) , the new Agent name
Output	JADE Management ontology kill action
Description	At this step the RA kills the illegal agents



**Figure 4.7** RA Step (4)

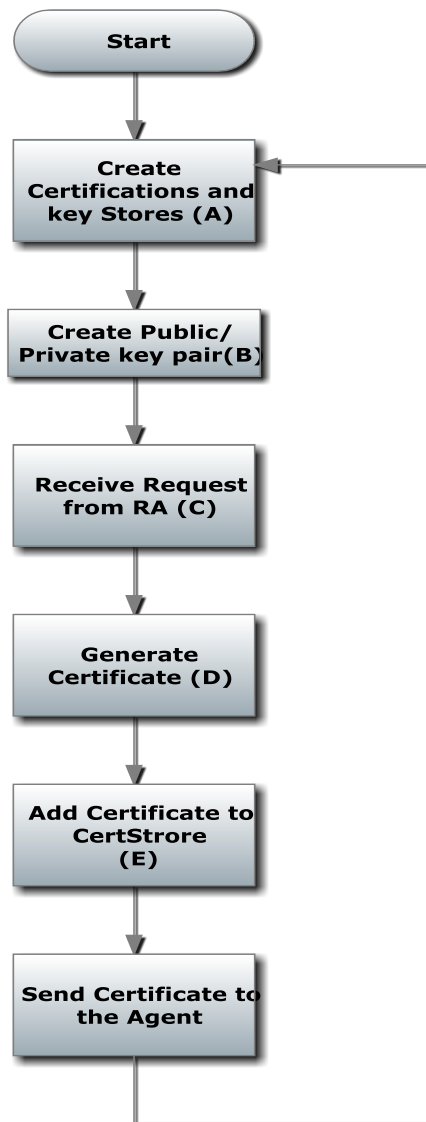
NAME:	Step(5)
Input	new Agent name
Output	ACLMessage to the CA
Description	At this step the RA sends the request (confirm) message to the CA to grant the new agent certificate



**Figure 4.8** RA Step (5)

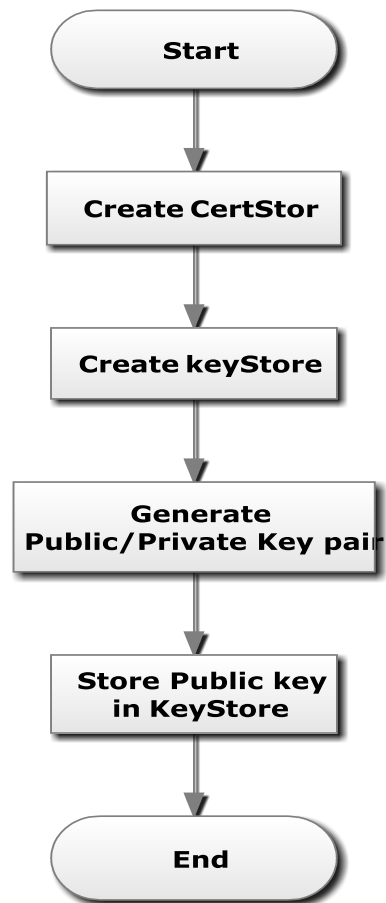
#### 4.4 .1.2 Certification Authority (CA)

This Agent responsible for certification in the platform and receives requests only from RA that after RA verify the request.



**Figure 4.9** CA Global Flowchart

NAME:	CA block A AND B
Input	ACLMessage
Output	ACLMessage , IOFiles (certificates and keys)
Description	Here the CA generate the public\private key pair and the directories keyStore and CertStore



**Figure 4.10:** CA block A and B

NAME:	CA Block C , D, E, F
Input	ACLMessage
Output	ACLMessage , IOFiles (certificates and keys)
Description	Here the CA receives the RA request message and generate the certificate store it in the certStore and send it to target agent



**Figure 4.11** CA block C, D, E, and F

#### 4.4 .1.3 Validation Authority (VA)

This agent validates the certificates that come from Container Guardian.

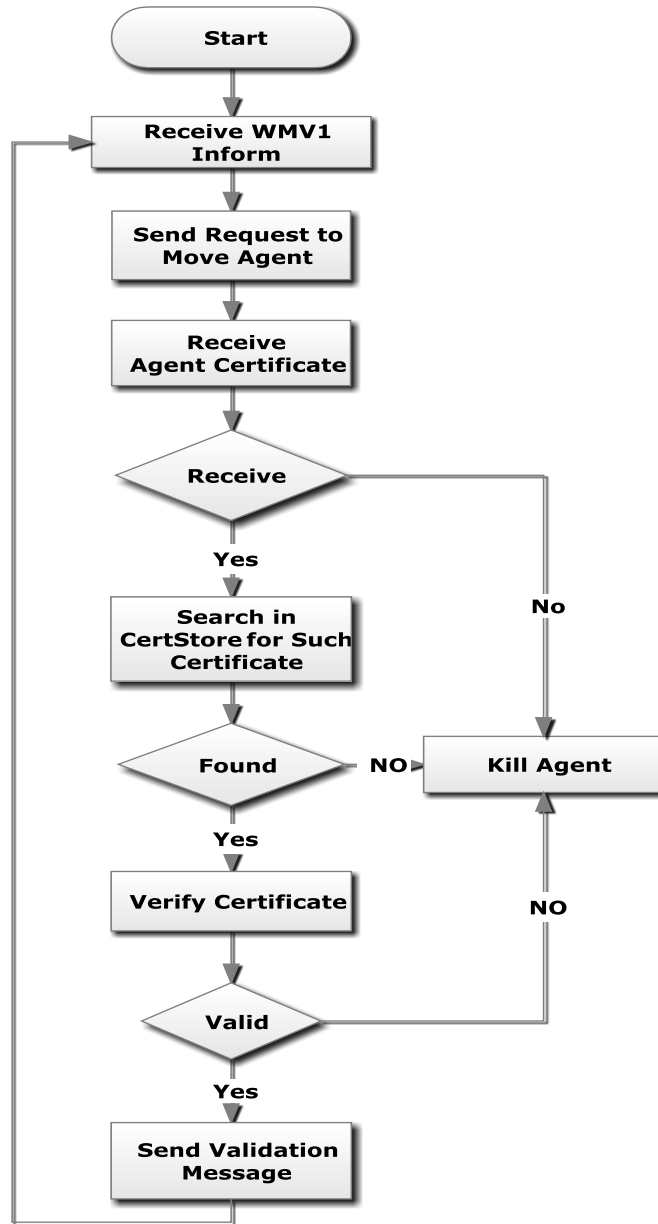
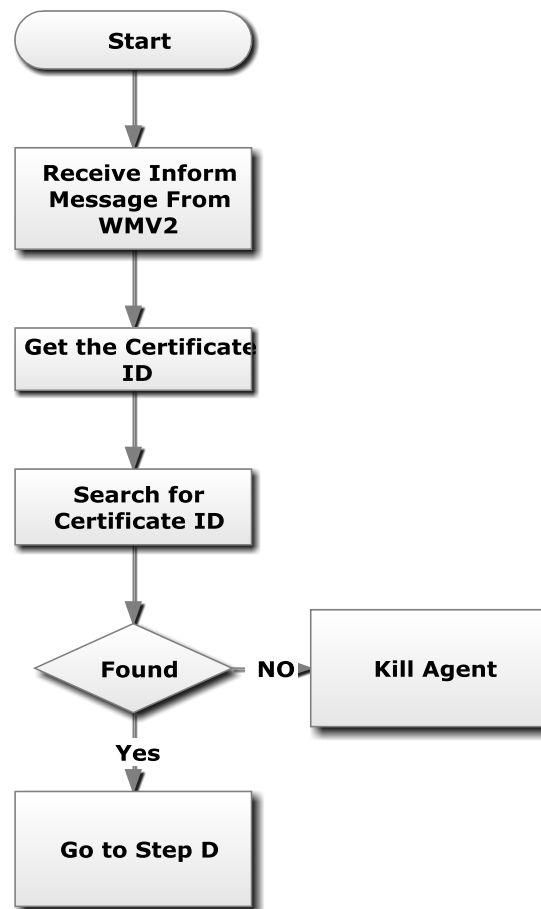


Figure 4.12 The VA Global

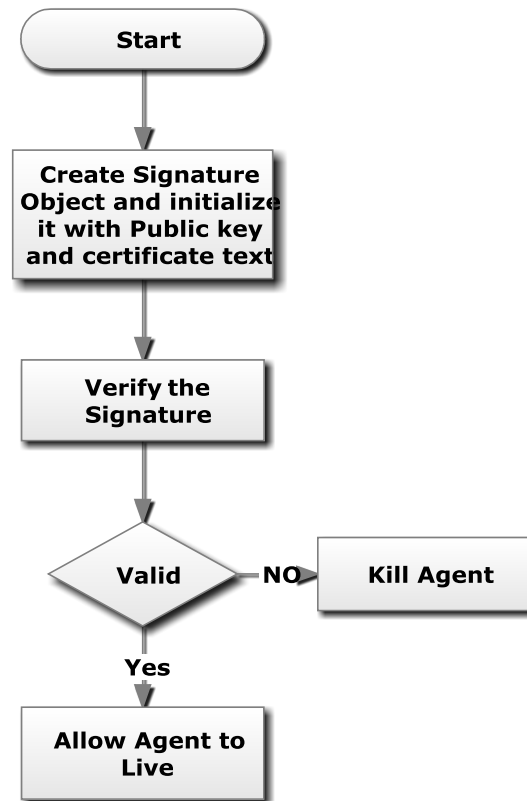


NAME:	VA Block A,B and C
Input	ACLMessage
Output	Boolean value , ACLMessage
Description	Here VA get the certificate id to search for it in the certStore if it found go to D if not send the invalid message



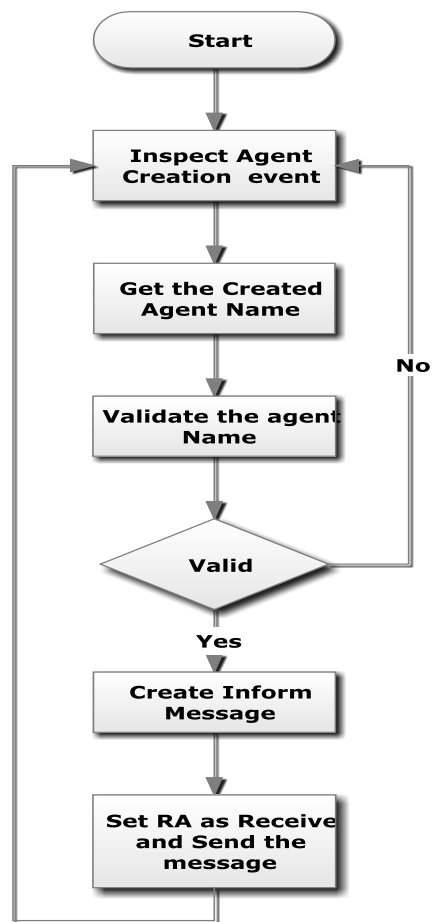
**Figure 4.13** VA Block A And B

NAME:	VA Block D and E
Input	Boolean, Agent name
Output	Kill ACLMessage
Description	Here VA validate the certificate and send back the response



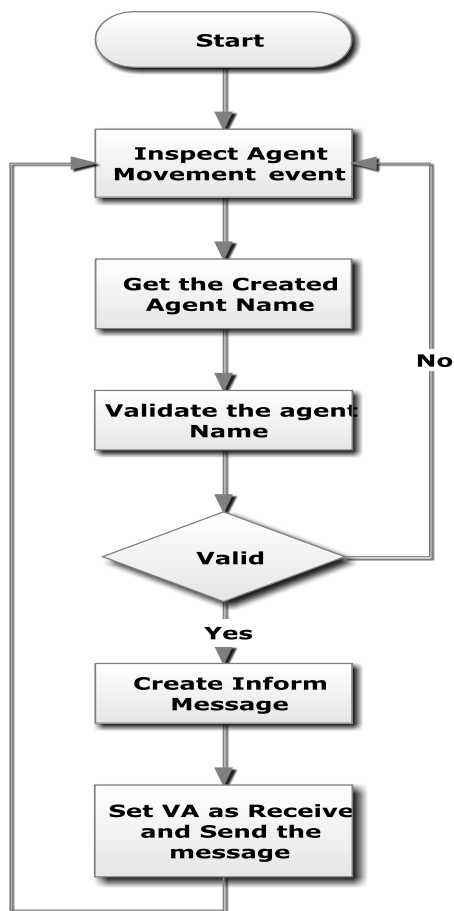
**Figure 4.14** VA Block D And E

NAME:	WMV1
Input	JADE introspection (Born Agent)
Output	RA inform ACLMessage
Description	Detailed flowchart for WMV1 the variable ev in the flowchart is referred to the event cached by the event handler



**Figure 4.15** WMV1 Flowchart

NAME:	WMV2
Input	JADE introspection (Move Agent)
Output	VA inform ACLMessage
Description	Detailed flowchart for WMV2 the variable ev in the flowchart is referred to the event cached by the event handler



**Figure 4.16** WMV2 Flowchart

## 4.5 Description of interaction classes

- **The Certificate class:** this class represents new data type called certificate that granted to agents that have illegal goals to live in JADE.
- **The Responder classes:** These classes represent the way that agent should response to JCOP agent's messages in order to live in the platform. Two classes defined in JCOP response package:
  - Certifier: A behavior that agent must use at the creation time to gain the certificate from CA and live in the platform.
  - Verifier: A behavior that agent use in the `afterMove()` method of class agent to allow agent to move legally between containers.

## 4.6 FNS Architecture

The File Name Searcher consist of the following components:

- Stationary Agent(RFA).
- Searcher Agent(SA).

The following figure shows the architecture of the system, including all components and their interactions

Detailed Specification of File Name Searcher components:

### Stationary Agent (RFA)

- Only one RFA agent allowed running over the distributed platform.
- When RFA created in the platform , it is call GetAvailableLocationsBehaviour which provide the RFA with available locations.

and call RecAgMsgBehaviour which wait until message arrive from (SA).

- RFA has user interface which can interact their user with (SA).

- when a new agent created the Search agent will lunch.
- when move method lunch the SA moving around all available locations.
- when a kill method lunch, the SA will die.
- it has a list shows the agents created (one agent on the platform)
- it has list shows the locations visited.

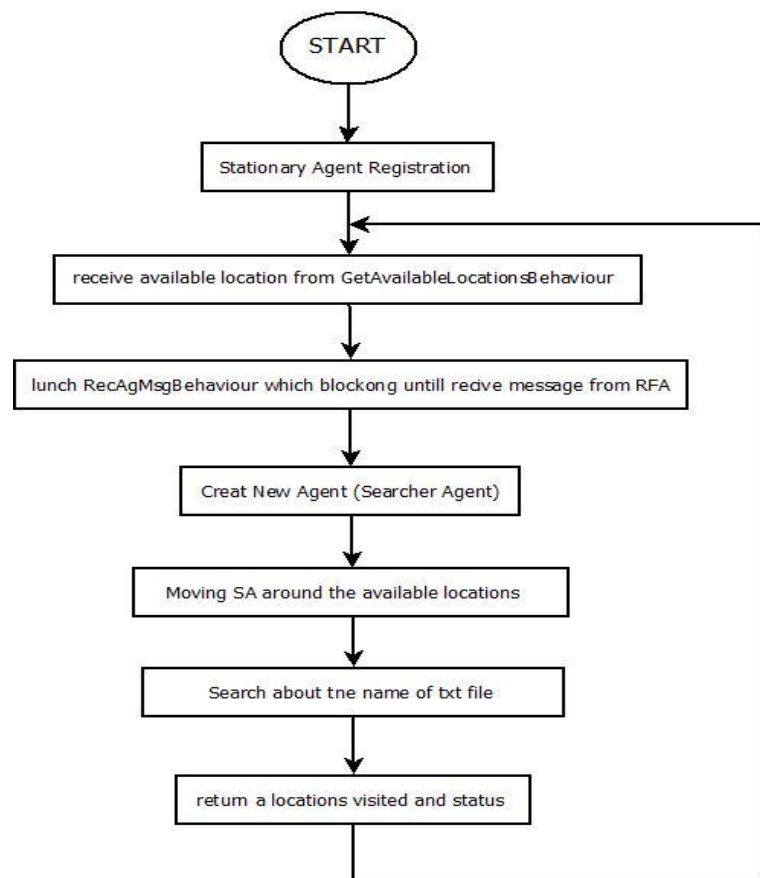
#### Searcher Agent (SA)

- Only one SA allowed running over the distributed platform.
- Search about txt file name.
- Moving around all available locations.

GUI unit contains the following components:

- 1- List for created agents.
- 2- List for visited locations.
- 3- button for created a new agent.
- 4- Button for move agent around all locations.
- 5- Button for killing agent.
- 6- Button for update locations.
- 7-Button for **quit**.

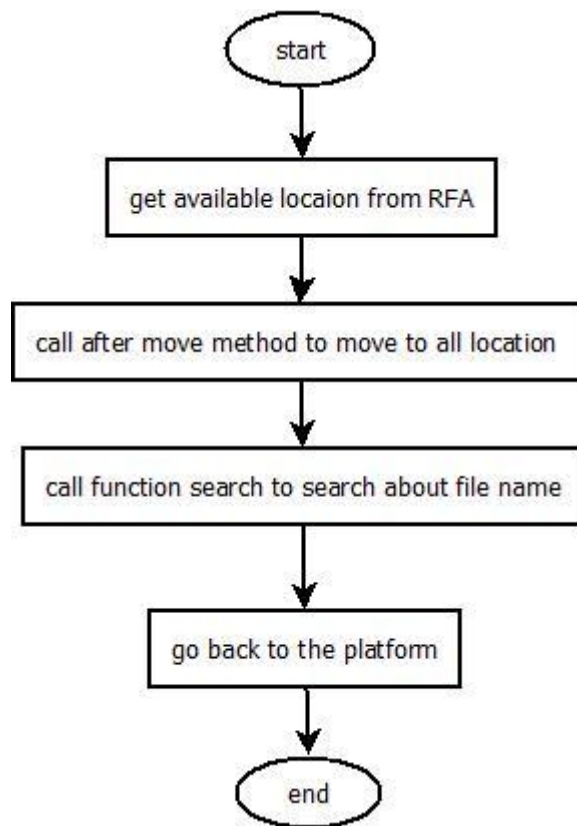
Name	FNS
Input	ACLMessage
Output	Result of searching
Description	The FNS application has two agents the first one is stationary agent which stay on the platform and the second one is mobile agent which is search about file name on distributed system and it has two behaviors one for locations and the another for receive messages.



**Figure 4.19FNS Flow Chart**

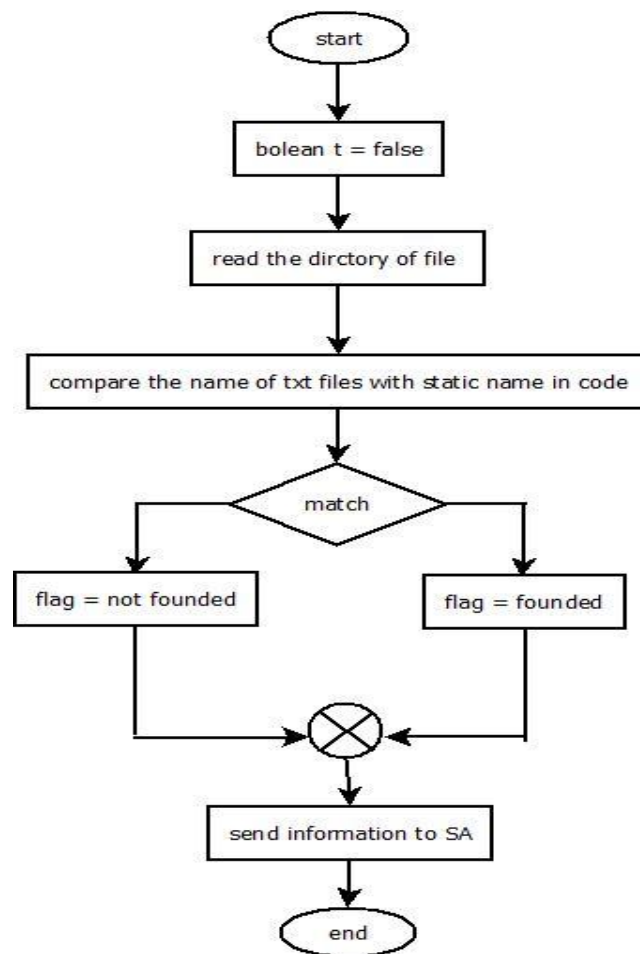


Nmae	Searcher agent
Input	File name
Output	Result and status
Description	This agent is mobile agent which has the file name to be search



**Figure 4.20** Searcher Agent Flowchart

Name	Search
Input	The directory where the file exist
Output	Flag
Description	This function search about file name in directory chosen



**Figure 4.20** Search Process Flowchart

## 4.7 Limitations

- This scheme assumes containers runs on JADE while JCOP is running trusted containers for trusted owners.
- The JADE platform gives relatively small number of agents (less than 100) so although JCOP does not add big overhead one memory and CPU but it decreases the number of agent's user can create on platform.
- In the JCOP design the fault tolerance was not issue.
- The JCOP agents are not fixed agents the removed easily by administrator and when they removed JADE can continue its work without any problems.

## 4.8 Assumptions

- The user Of JCOP assumed to be familiar with programming.
- The containers that connected to the platform assumed to be legitimate ones.
- All JCOP agents assumed to stay a life during as long as the JADE platform running.
- Every time JCOP is started it will override the certificates in the certificate store.
- The user of FNS assumed to be familiar with computers.
- The shared folder that the FNS searched in assumed to be previously created.

## **4.9 Summary**

In this chapter a detailed description of the design of the different system components and modules, Including system architecture, flowcharts, assumptions and limitations.

# Chapter 5

# IMPLEMENTATION AND TESTING

**5.1 Development Environment**

**5.2 Development Process**

**5.3 Testing**

**5.4 Summary**

## **Chapter Five**

### **Implementation and Testing**

In this chapter description of how components of the system were implemented, and also provide the system testing process.

#### **5.1 Development Environment**

The implementation of the system was done on two environments, since JADE (as a simulator) can be used with full functionality on one computer most of the implementation and software components testing was made on one computer, however, to become closer to the real usage of the system, rich part of the testing was done over a local area network

**The software that was used in the implementation and testing phases consist of:**

- JADE package: this package should be installed on the working computer and the testing computers as well, in order to start the different JADE containers.
- JDK: should be installed on the working computers, in order to compile java source code into byte code, and then interpreted as machine code to become executable also for the testing computers it should be installed as well, in order to make JADE containers runnable on the test computers.
- Net Beans IDE: a free and very helpful IDE that us ready GUI components that facilitated the design and implementation of the system we use this IDE to design the GUI, and to write the source code of all functions.

## **5.2 Development Process**

Spiral software engineering model was used in the development process, because the overall features and requirements of the system was not clear at the beginning of the development process, many prototypes of the system was implemented, refinement of the requirements was done, and components was tested individually and many integrity tests was done, until a final version of the system was delivered.

### **5.2.1 Phases of System Development:**

#### ***Phase 1: Development of JCOP***

- Implementation of the basic JCOP agents.
- Implementation of the agent's behaviors.
- Implementation of the JCOP reusable package classes.

#### ***Phase2: Development of FNS system***

- Implementation of the basic FNS agent behaviors.
- Implementation of the basic FNS agents.
- Implementation of the agents GUI.

#### ***Phase3: Reimplementation of FNS over JCOP (Integration).***

- Perform modifications to FNS to integrate it with JCOP.

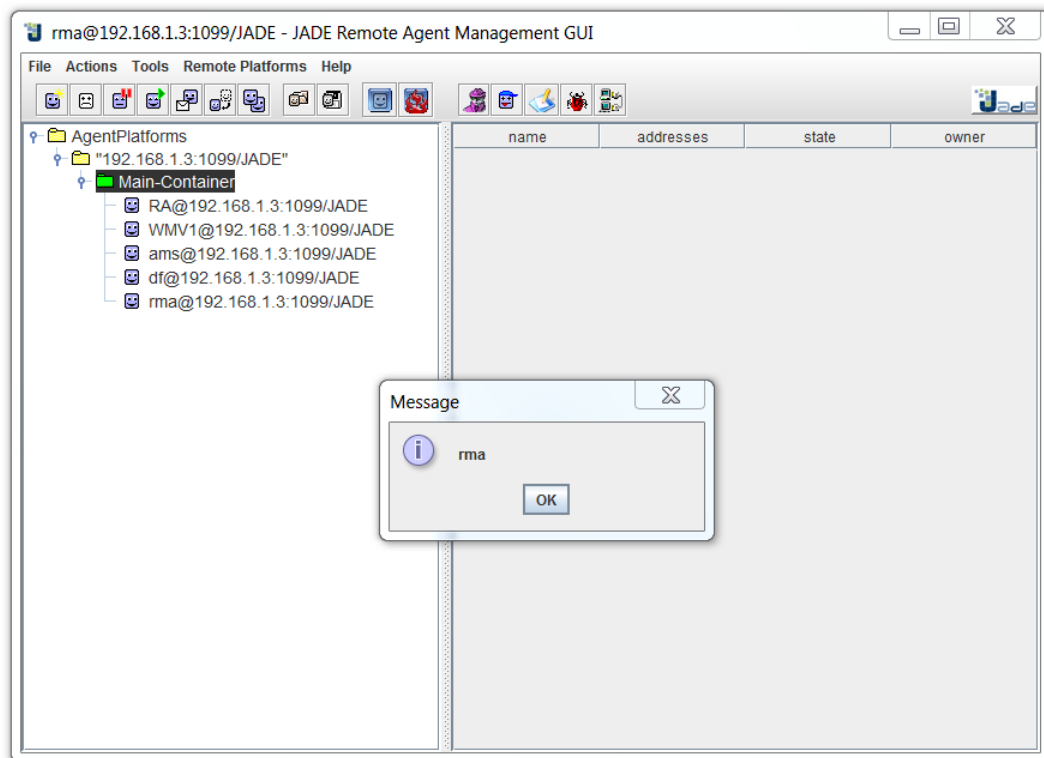
## 5.3 Testing

The following figures show the results of different inputs to the system.

Figures describe:

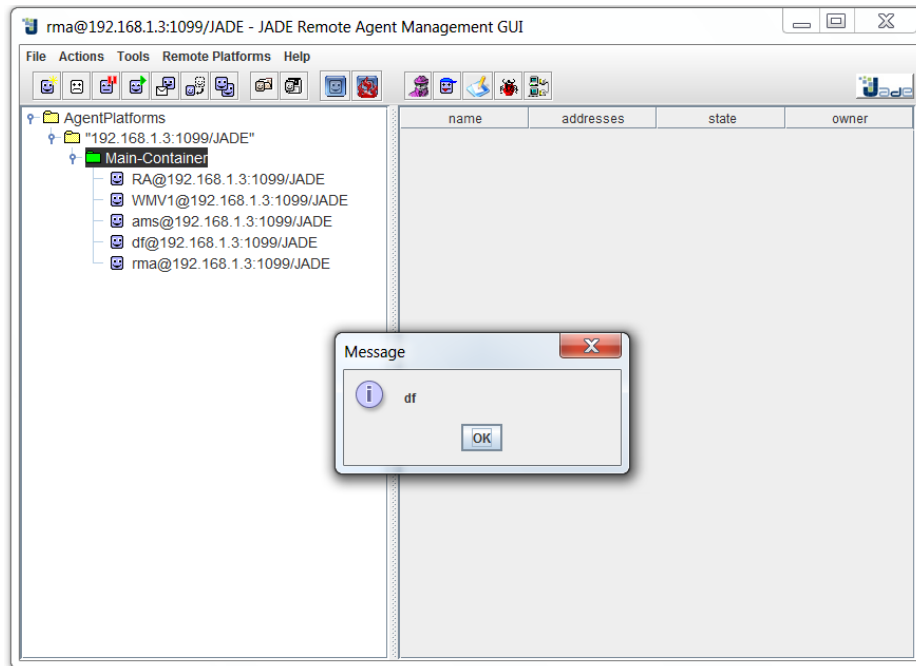
- JCOP underlying message exchange with agents runs over JADE.
- FNS working over JCOP.

**JCOP underlying message exchange with agents runs over JADE.**

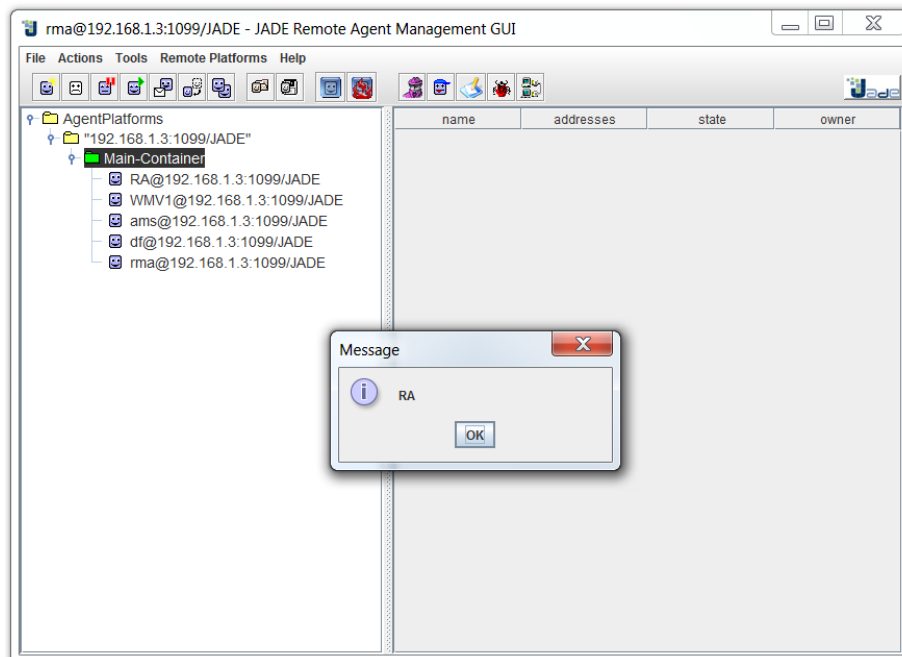


**Figure 5.1** WMV1 show the name rma after discovering it.

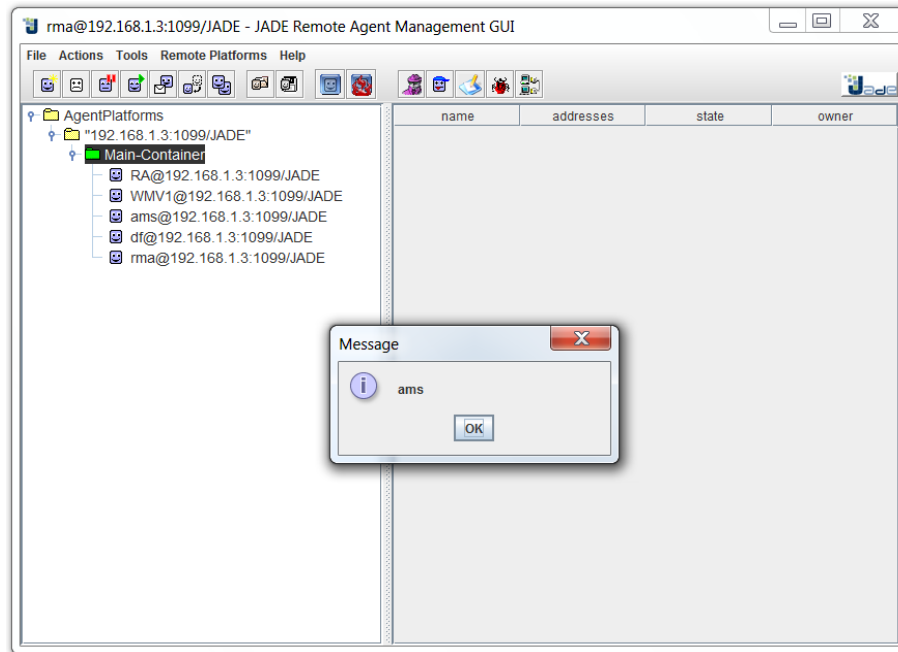




**Figure 5.2** WMV1 show the name df after discovering it.



**Figure 5.3** WMV1 show the name RA after discovering it.



**Figure 5.4** WMV1 show the name ams after discovering it.

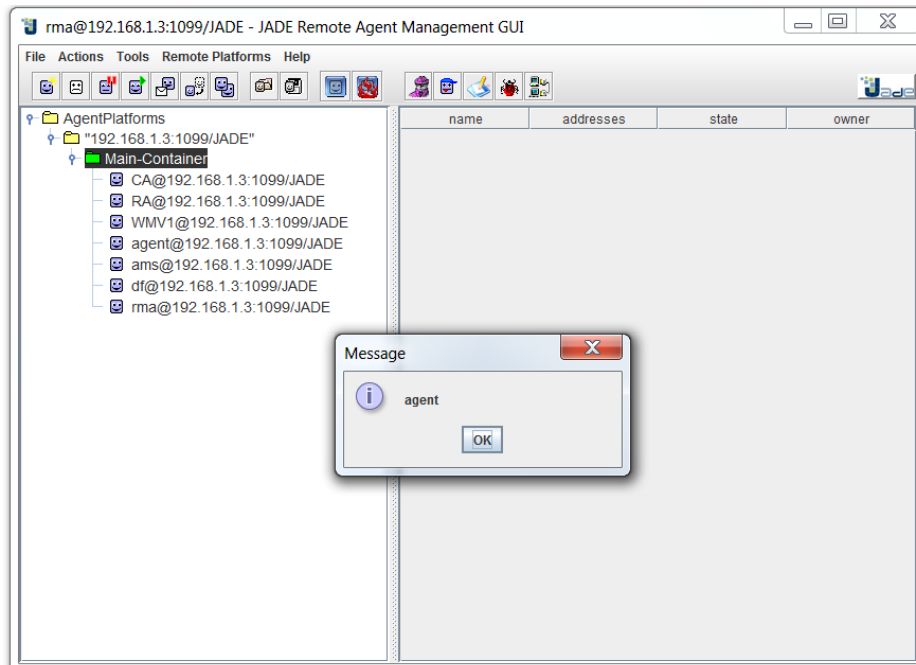
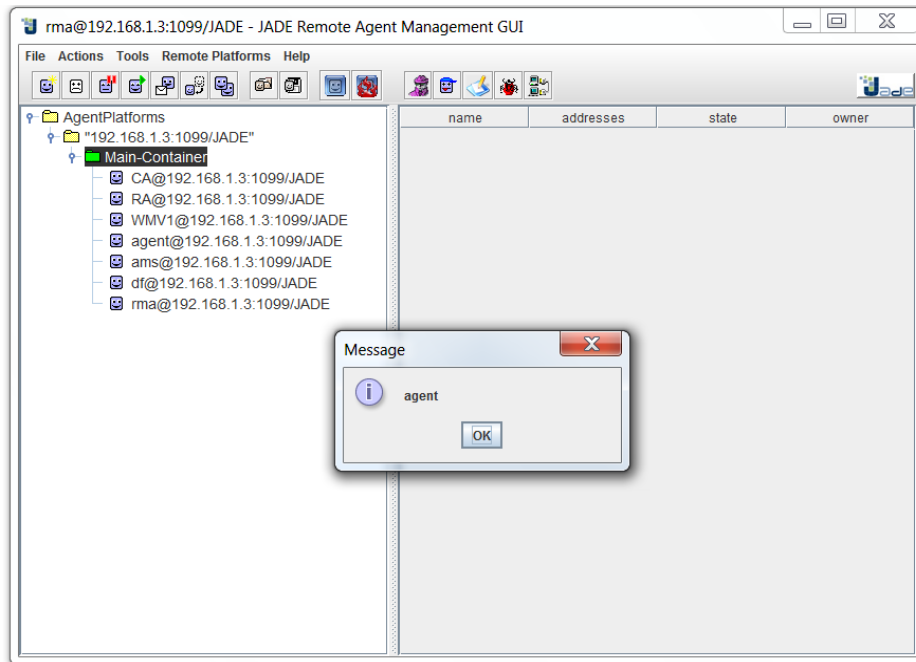
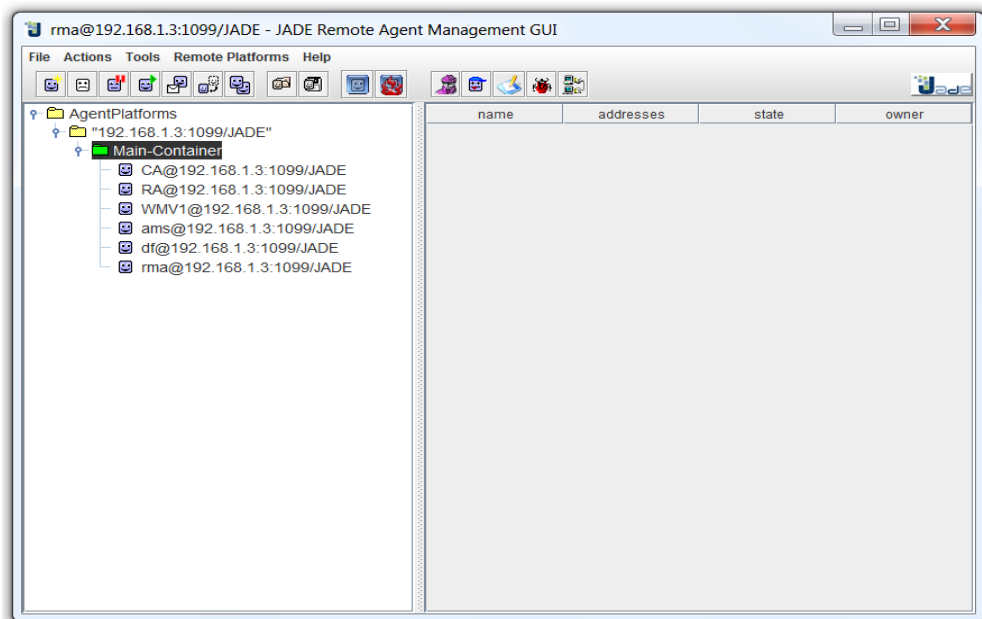


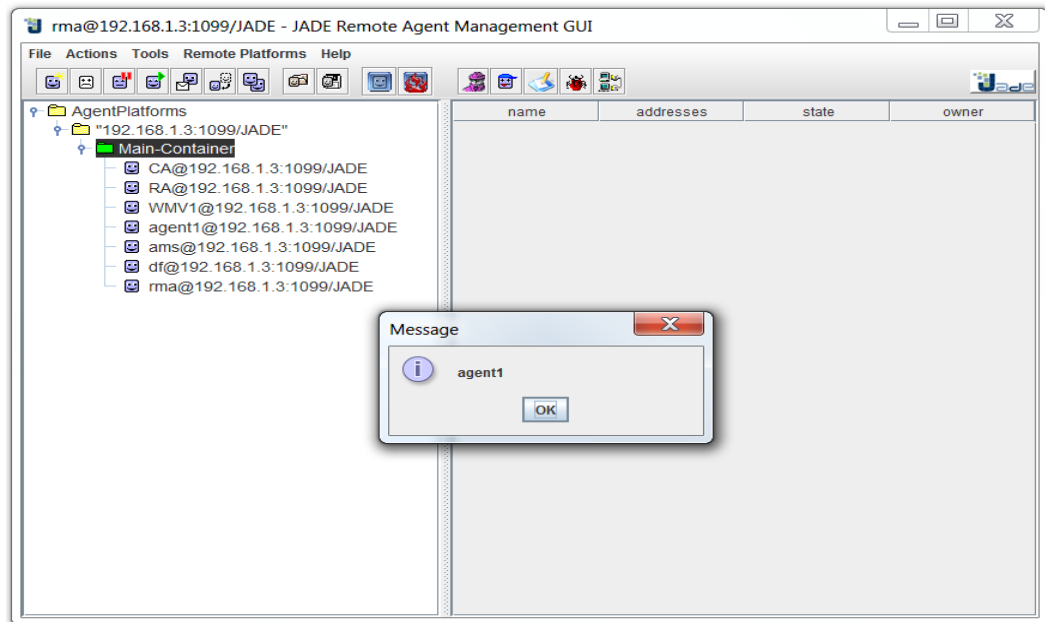
Figure 5.6 WMV1 show the name of new agent called “agent” after discovering it.



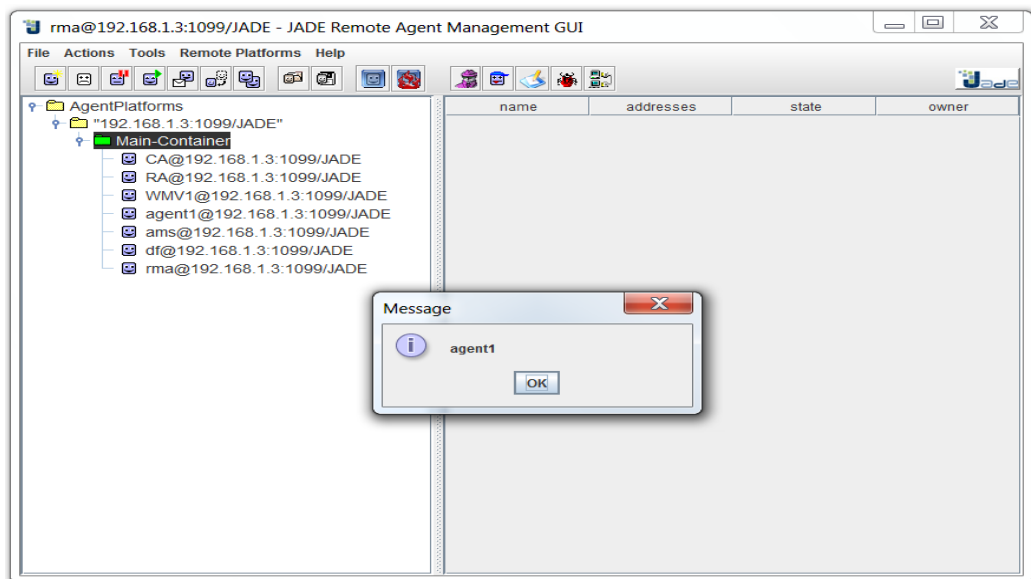
**Figure 5.6** RA show content of a message sent by WMV1 carrying the name of the new agent (this agent did not follow the JCOP restrictions).



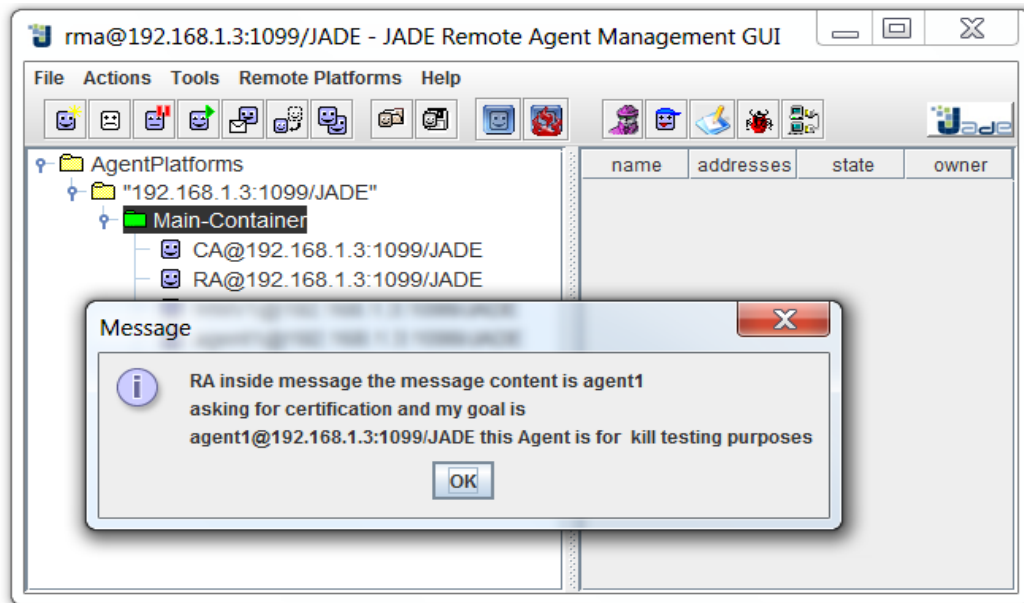
**Figure 5.7** RA kills the new agent "agent".



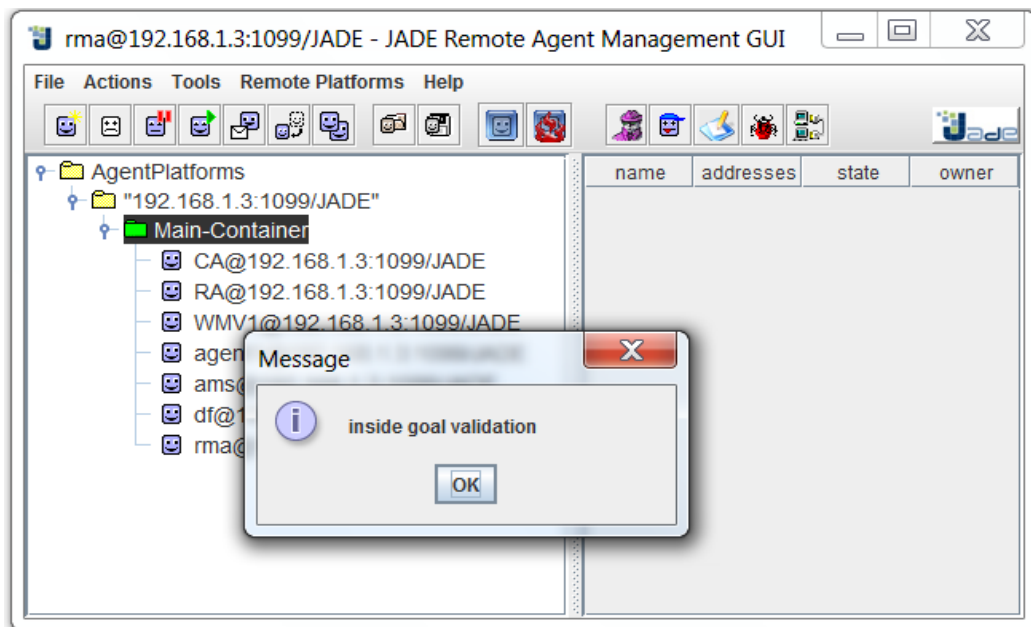
**Figure 5.8** WMV1 show the name of new agent called “agent1”  
after discovering it.



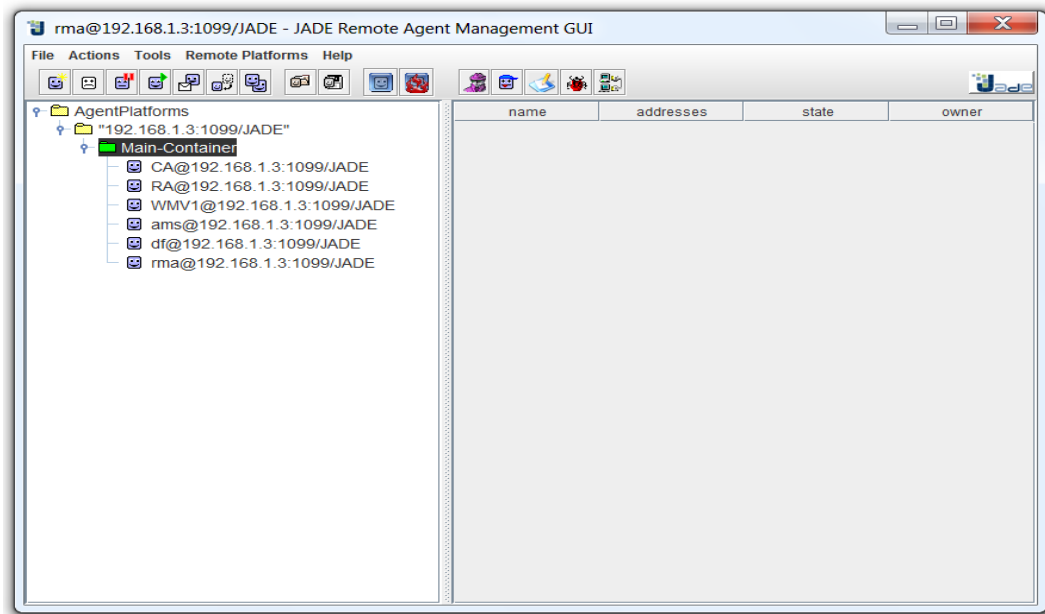
**Figure 5.9** RA show content of a message sent by WMV1 carrying the name of the new agent (this agent follow the JCOP restrictions but with illegal goal).



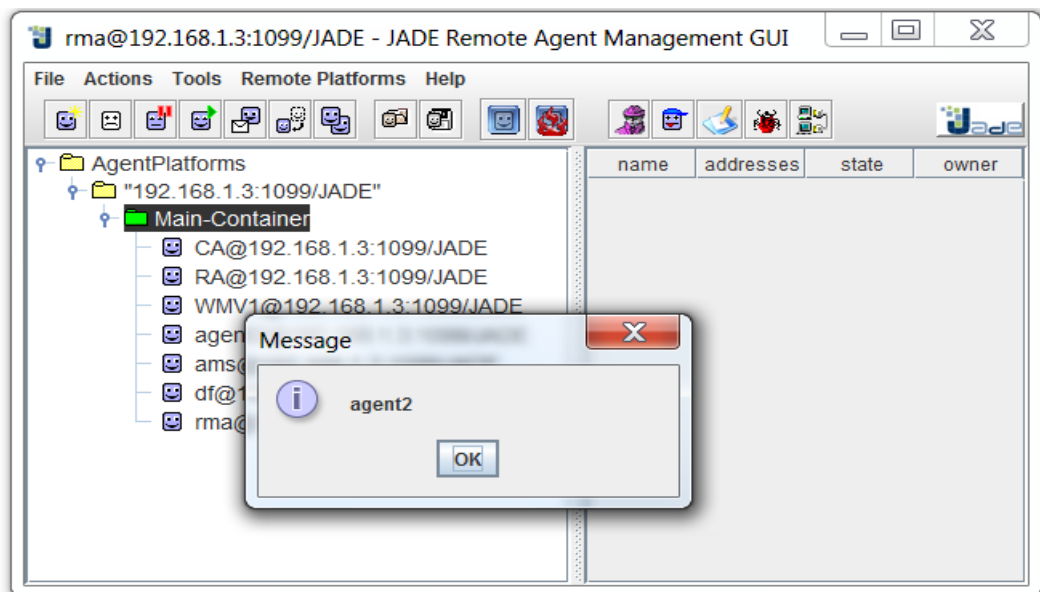
**Figure 5.10** a message box that shows the content of agent request message (notice the word kill in the message content where kill is restricted word)



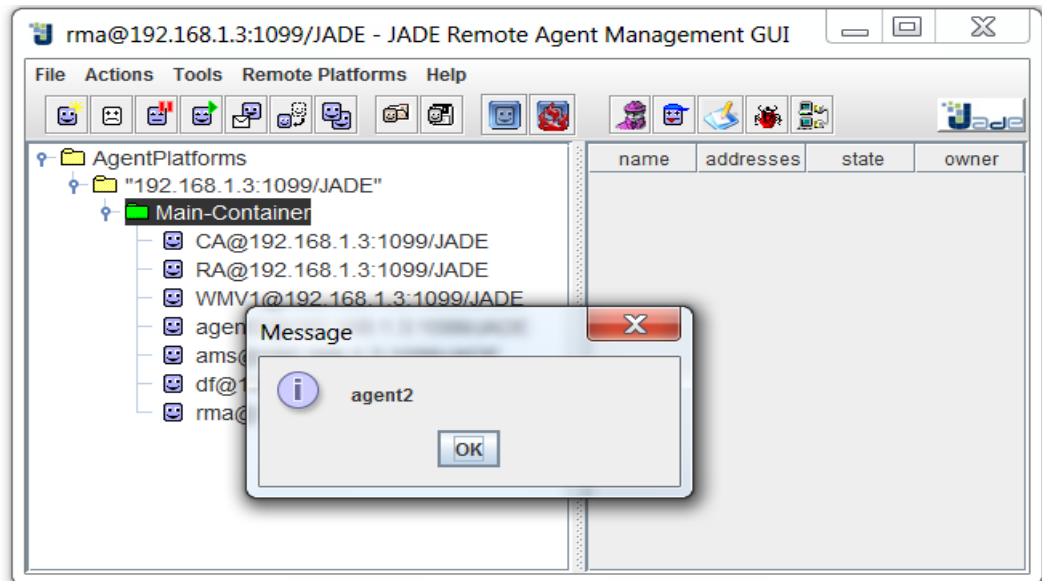
**Figure 5.11** a message box shows the request message inside the text validation process.



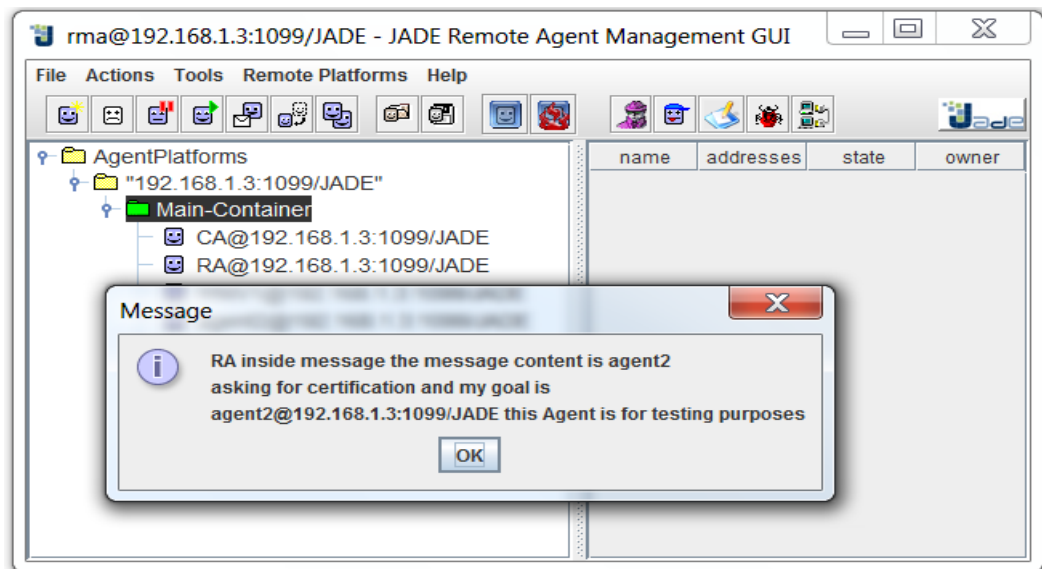
**Figure 5.12** RA kills the new agent “agent1” because it has illegal goal.



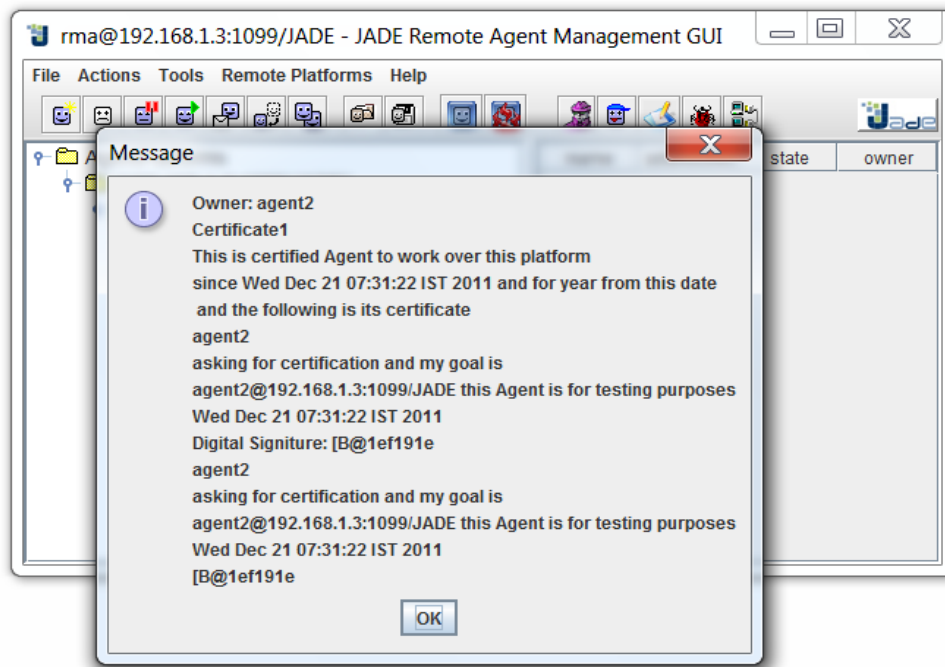
**Figure 5.13** WMV1 show the name of new agent called “agent2” after discovering it.



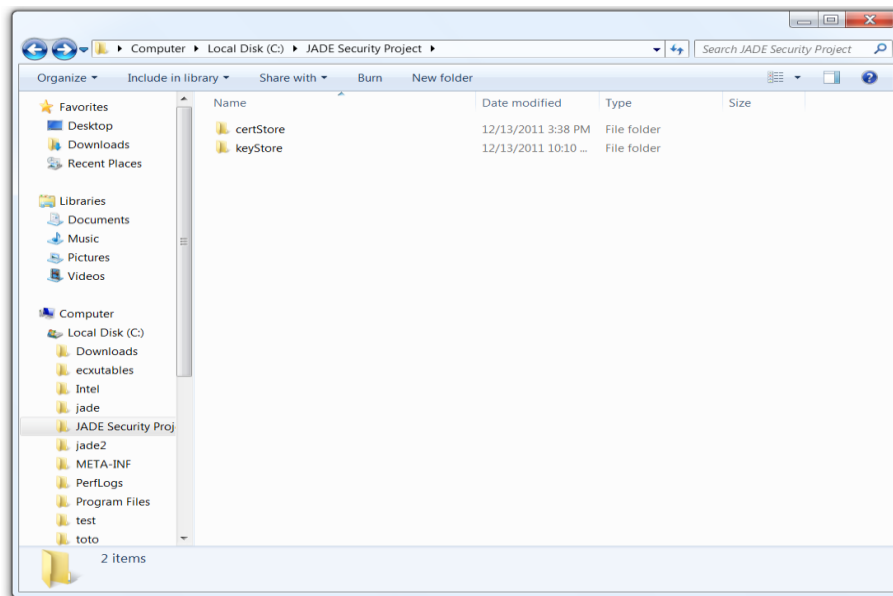
**Figure 5.14** RA show content of a message sent by WMV1 carrying the name of the new agent (this agent follow the JCOP restrictions).



**Figure 5.15** a message box that shows the content of agent request message (notice that the message did not contain any restricted word)

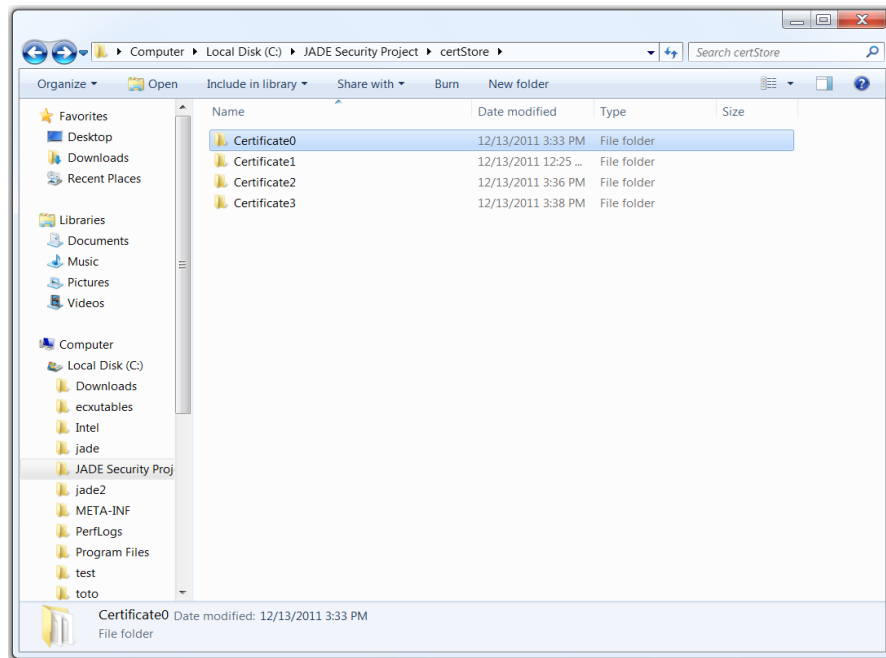


**Figure 5.16** a message box shows the certification granted from CA to new agent.

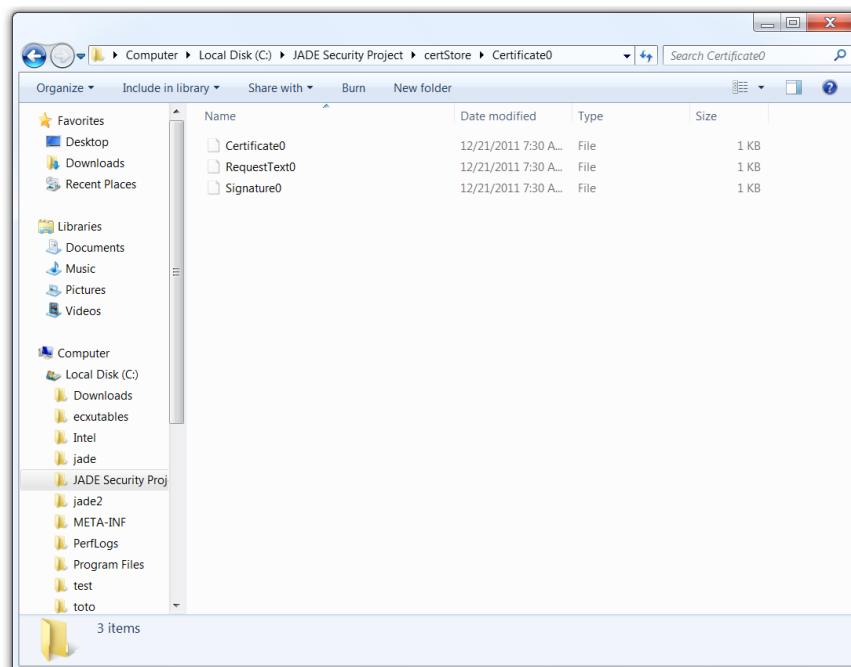


**Figure 5.17** the JCOP certStore and keyStore directories.

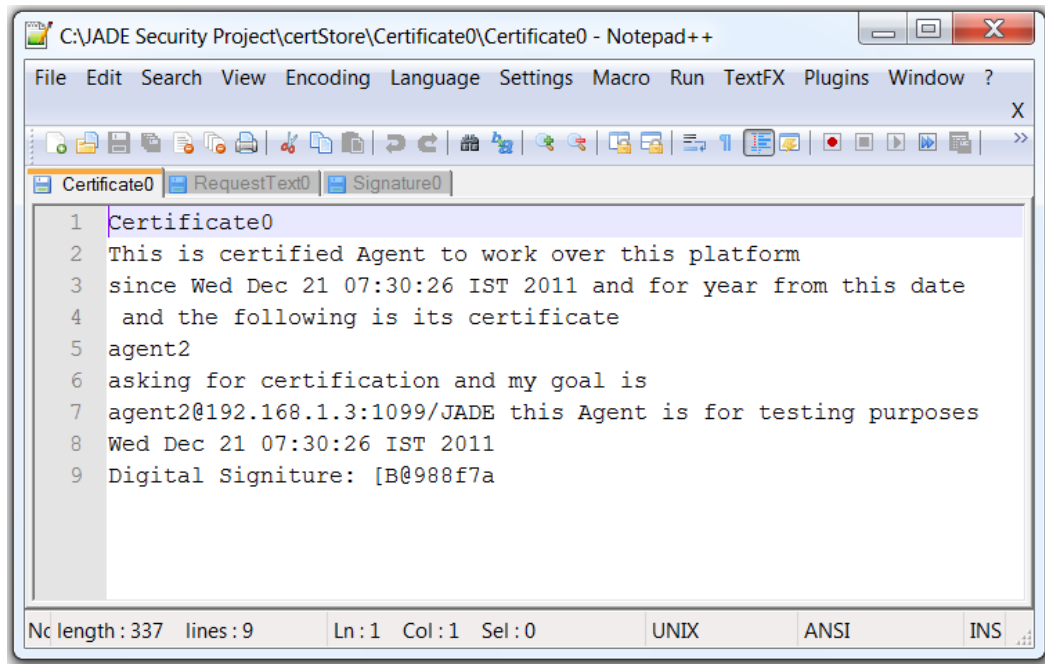




**Figure 5.18** inside JCOP certStore directory.



**Figure 5.19** inside JCOP Certificate0 directory.

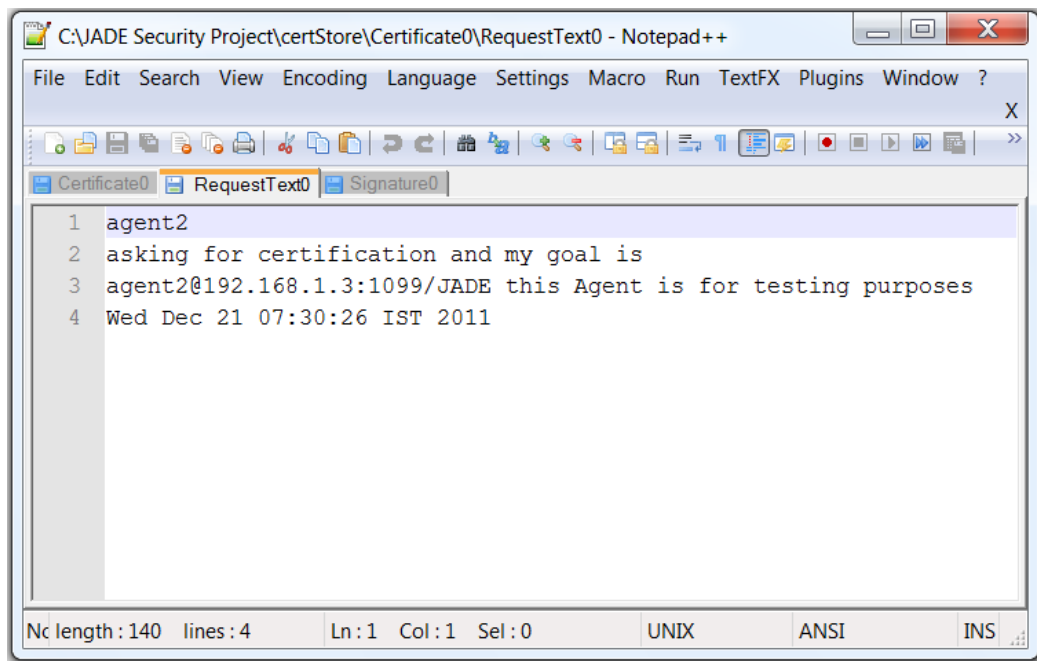


A screenshot of a Notepad++ window titled "C:\JADE Security Project\certStore\Certificate0\Certificate0 - Notepad++". The window displays the content of the "Certificate0" file. The text is as follows:

```
1 Certificate0
2 This is certified Agent to work over this platform
3 since Wed Dec 21 07:30:26 IST 2011 and for year from this date
4 and the following is its certificate
5 agent2
6 asking for certification and my goal is
7 agent2@192.168.1.3:1099/JADE this Agent is for testing purposes
8 Wed Dec 21 07:30:26 IST 2011
9 Digital Signature: [B@988f7a
```

The status bar at the bottom indicates "No length : 337 lines : 9", "Ln : 1 Col : 1 Sel : 0", and encoding "UNIX ANSI INS".

**Figure 5.20** inside JCOP Certificate0 file.

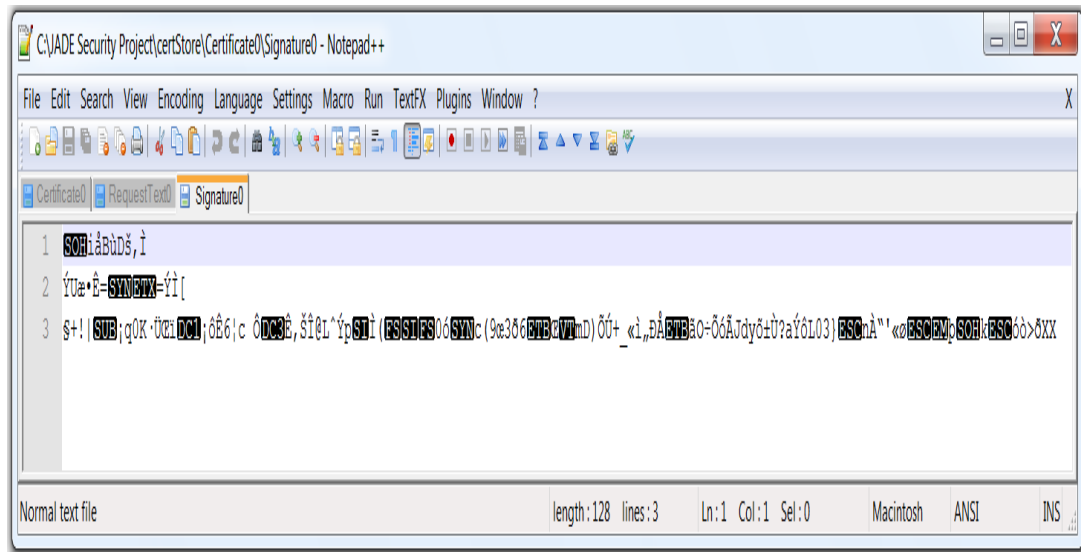


A screenshot of a Notepad++ window titled "C:\JADE Security Project\certStore\Certificate0\RequestText0 - Notepad++". The window displays the content of the "RequestText0" file. The text is as follows:

```
1 agent2
2 asking for certification and my goal is
3 agent2@192.168.1.3:1099/JADE this Agent is for testing purposes
4 Wed Dec 21 07:30:26 IST 2011
```

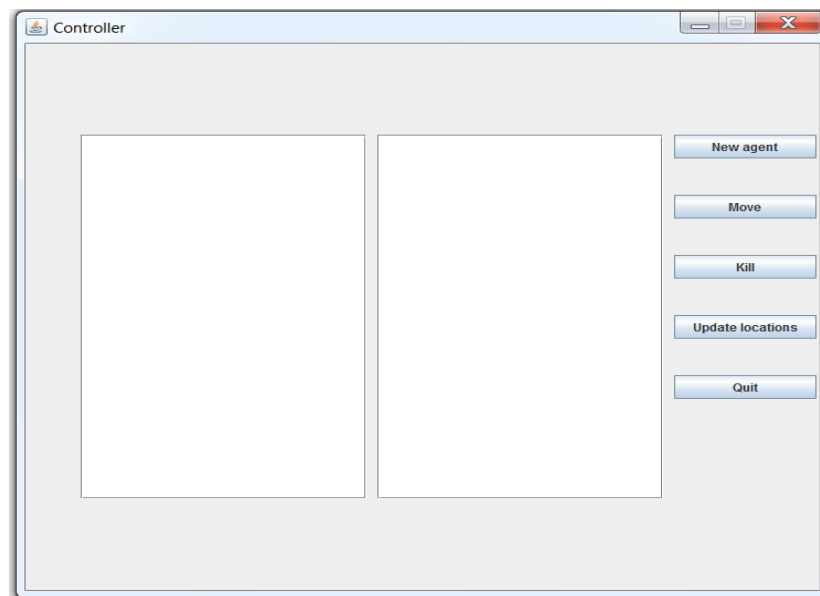
The status bar at the bottom indicates "No length : 140 lines : 4", "Ln : 1 Col : 1 Sel : 0", and encoding "UNIX ANSI INS".

**Figure 5.21** inside JCOP RequestText0 file.



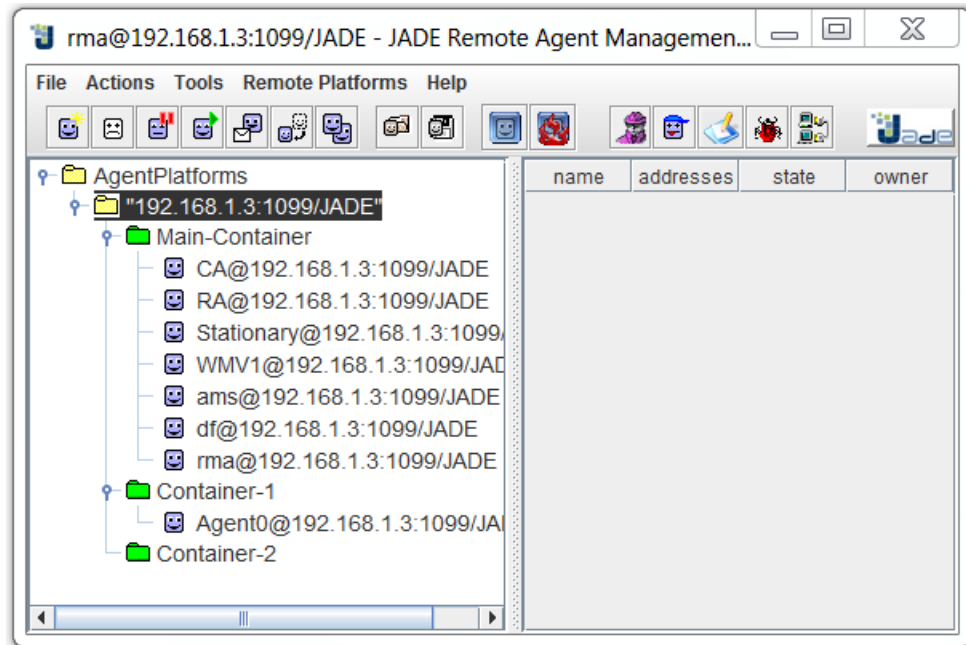
**Figure 5.22** inside JCOP Signature0 file.

- **FNS working over JCOP.**



**Figure 5.23** the FNS stationary agent “Controller”.





**Figure 5.22** the searcher agent search in machine 1.

## 5.4 Summary

In this chapter description of how the components of the system were implemented, testing of the system, development process and phases.

# **Chapter 6**

# **Conclusion**

# **&**

# **future work**

6.1 Experimental results

6.2 Conclusions

6.3 Future Work

6.4 Summery

## Chapter Six

### Conclusion and Future Work

In this chapter adscription of the experiments that were done, conclusions concluded from this working on the project, future work and improvements to be done on the system.

#### 6.1 experimental Results

In this section we show different experiments results (outputs and memory and CPU load).

**Table 6.1** Hardware used in the experiments

Experiment 1	Computer1: Corei3, 3.07 GHZ, 6GB RAM
Experiment2	Computer1: Corei3, 3.07 GHZ, 6GB RAM
Experiment3	Computer1:Core2 duo, 2.2 GHz, 4GB RAM Computer2:Centrino Dual, 1.6 GHz, 2GB RAM

Key terms used in the context of the comparison in experiments:

#### For Memory:

- Commit: amount of virtual memory reserved by the operating system for the process in (KB).
- Working Set: amount of physical memory currently in use by the process in (KB).

- Shareable: amount of physical memory in use by the process that can be shared with other processes in (KB).
- Private: amount of physical memory in use by the process that cannot be used by other processes in (KB).

**For CPU:**

- Threads: number of active threads.
- CPU:s current percentage of CPU consumption by process(in this experiment the maximum CPU is taken)
- Average CPU: average percentage of CPU consumption by the process in (60 seconds) (in this experiment the maximum average CPU is taken).

### **6.1.1 Experiment One**

The outputs of this experiment matching the expected results RA kill all illegal agents (those who did not implement JCOP classes and those who have illegal goals) and CA grant all legal agents certificates. (In this experiment run only JCOP and simple agents FNS system was not tested in this experiment)

**Table 6.2** Experiment 1 Memory Comparison



Process	JCOP ON/OFF	Commit (KB)	Working Set (KB)	Private (KB)	Private (KB)
JADE	OFF	52.484	48.260	18.260	30.012
JADE	ON	55.440	60.124	18.372	41.752

**Table 6.2** Experiment 1 CPU Comparison

Process	JCOP ON/OFF	Threads	CPU	Average CPU
JADE	OFF	31	2	0.75
JADE	ON	35	4	0.85

A remarkable note here is that even if JCOP agents are not all in duty but it stills add small amount of overhead to the system performance. And this is proving the promising future of JCOP.

### 6.1.2 Experiment Two

The outputs of this experiment matching the expected results RA kill all illegal agents (those who did not implement JCOP classes and those who have illegal goals) and CA grant all legal agents certificates. (In this experiment both JCOP and FNS system are tested together).

**Table 6.3** Experiment 2 Memory Comparison

Process	JCOP	Commit	Working Set	Private (KB)	Private (KB)
---------	------	--------	-------------	--------------	--------------

	ON/OFF	(KB)	(KB)		
JADE	OFF	52.484	48.260	18.260	30.012
JADE	ON	64.296	63.624	18.324	45.312

**Table 6.4** Experiment 2 CPU Comparison

Process	JCOP ON/OFF	Threads	CPU	Average CPU
JADE	OFF	31	2	0.75
JADE	ON	41	25	24.90

The results of this experiment are clearly shows that the FNS system adds big load to memory and CPU and specially CPU.

### 6.1.3 Experiment Three

This experiment is like experiment tow but it's executed over a network.

**Table 6.5** Experiment 3 Memory Comparison

Process	JCOP ON/OFF	Commit (KB)	Working Set (KB)	Private (KB)	Private (KB)
JADE	OFF	83.144	73.746	21.080	52.684
JADE	ON	83.572	74.132	21.080	53.052

**Table 6.6** Experiment 4 CPU Comparison

Process	JCOP ON/OFF	Threads	CPU	Average CPU
JADE	OFF	31	0	0.73
JADE	ON	38	0	0.88

This Experiment results are not hundred percent correct because the FNS behavior still confused. But memory consumption clearly appears greater because of the connection overhead.

## 6.2 Conclusion

From the work shown above we conclude the follow:

- JCOP is good security framework for but it still has some dark sides that are not clearly verified.
- The security is not a joke it is time consuming and highly cost process but in some cases security is note optional.
- Security to be truly successful it must be integrated inside the initial design of the system.

### **6.3 Future Work**

The future work for JCOP is huge because this project is only the kernel (it is only the start). And here some trends in advancing JCOP:

- Implementing one fat agent that gathers all the SM agents.
- Considering fault tolerance issue in the design.
- Make the JCOP agent fixed on platform.
- Making the certificate life time shorter.
- Secure the CertStore and the KeyStore.

### **6.4 Summery**

This chapter explains the experiments that were done, conclusions concluded form working on the project, conclusions from the experiments, and future work and improvements on the system.

## References:

1. Roberto Silveira Silva Filho Department of Information and Computer Science, University of California, Irvine “The Mobile Agents Paradigm.”  
(<http://awareness.ics.uci.edu/~rsilvafi/papers/SoftwareEngineeringFinalPaper.pdf>)
2. Louise Eggöy, Hanna Kostmann. 18 June 2001 Communication methods using distributed systems. Department of Information Technology Media Technology program, 180 credits Master thesis, 20 credit,  
(<http://apachepersonal.miun.se/~loveke/magister/uppsatser/original.pdf>)
3. [S. Venkatesan](#), [P. Dhavachelvan](#), [T. Vengattaraman](#) , [C. Chellappan](#), [Anurika Vaish](#) November, 2010 “Advanced mobile agent security models for code integrity and malicious availability check” Journal of Network and Computer Applications [archive](#) Volume 33 Issue6.  
([Advanced mobile agent security models for code integrity and malicious availability check](#))
4. A Buddy Model of Security for Mobile Agent Communities Operating in Pervasive Scenarios John Page, Arkady Zaslavsky, Maria Indrawan School of Computer Science and Software Engineering Monash University, Melbourne, Australia. ([A buddy model of security for mobile agent communities operating in pervasive scenarios](#))
5. [Abdelhamid Ouardani](#) , [Samuel Pierre](#) , [Hanifa Boucheneb](#), August, 2007. “A security protocol for mobile agents based upon the cooperation of sedentary agents” Journal of Network and Computer Applications [archive](#) Volume 30 Issue 3([A security protocol for mobile agents based upon the cooperation of sedentary agents](#))

6. Sheng Zhong, Yang Richard Yang Verifiable Distributed Oblivious Transfer and Mobile Agent Security (Sheng Department of Computer Science Yale University New Haven, CT 06520-8285 [sheng.zhong@yale.edu](mailto:sheng.zhong@yale.edu) ) (Yang Department of Computer Science Yale University New Haven, CT 06520-8285 [yry@cs.yale.edu](mailto:yry@cs.yale.edu) ) ([http://www.google.ps/url?sa=t&source=web&cd=1&ved=0CBoQFjAA&url=http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.10.6628%26rep%3Drep1%26type%3Dpdf&rct=j&q=Verifiable%20Distributed%20Oblivious%20Transfer&ei=53GbTaSeFcuXOp\\_graAH&usg=AFQjCNFEy\\_4CFdDk92oRKeLo0MDT9V1MtA](http://www.google.ps/url?sa=t&source=web&cd=1&ved=0CBoQFjAA&url=http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.10.6628%26rep%3Drep1%26type%3Dpdf&rct=j&q=Verifiable%20Distributed%20Oblivious%20Transfer&ei=53GbTaSeFcuXOp_graAH&usg=AFQjCNFEy_4CFdDk92oRKeLo0MDT9V1MtA))
7. Ke Xu, B.E. May 2004 Mobile Agent Security Through Multi-Agent Cryptographic /Protocols UNIVERSITY OF NORTH TEXAS May 2004 (<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.97.338&rep=rep1&type=pdf> ) ,Department of Information Technology Media Technology program, 180 credits Master thesis, 20 credit.
8. Hiroshi Matsuno" Implementing Mobile Agents in Genome Information Processing "Faculty of Science, Yamaguchi University, 1677-1 Yoshida, Yamaguchi, 753, Japan.
9. O.Univ.Prof. Dipl.-Ing. Dr.techn. Mehdi Jazayeri , **Building Secure Mobile Agents** ,The Supervisor-Worker Framework.
10. Anselm Lingnau, An HTTP-based Infrastructure for Mobile Agents.
11. NIST Special Publication 800-19 – Mobile Agent Security.
13. Computer security Principals and practice by WILLIAM STALLINGS & LAWRIE BROWN. Chapter 7 page 250.
- 14 <http://searchsecurity.techtarget.com/sDefinition/0,,sid14gci498695,00.html>

- 15 Computer security Principals and practice by WILLIAM STALLINGS & LAWRIE BROWN. Chapter 1 page 16.
- 16 [https://my.tennessee.edu/portal/page?\\_pageid=40,614533&\\_dad=portal&\\_schema=PORTAL](https://my.tennessee.edu/portal/page?_pageid=40,614533&_dad=portal&_schema=PORTAL).
- 17 <http://en.wikipedia.org/wiki/Eavesdropping> .
- 18 [http://en.wikipedia.org/wiki/Man-in-the-middle\\_attack](http://en.wikipedia.org/wiki/Man-in-the-middle_attack) .
- 19 Computer security Principals and practice by WILLIAM STALLINGS & LAWRIE BROWN. Chapter 1 page 19.
- 20 <http://JADE.tilab.com/doc/tutorials/JADEAdmin/JADEArchitecture.html>
- 21 **Developing multi-agent systems with JADE** By Fabio Luigi Bellifemine, Giovanni Caire, Dominic Greenwood
- 22 [http://download.cnet.com/Java-Runtime-Environment-JRE/3000-2356\\_4-10009607.html](http://download.cnet.com/Java-Runtime-Environment-JRE/3000-2356_4-10009607.html)
- 23 <http://download.oracle.com/javase/6/docs/>
- 24 [http://www.sans.org/reading\\_room/whitepapers/application/distributed-systems-security-java-corba-com-plus\\_28](http://www.sans.org/reading_room/whitepapers/application/distributed-systems-security-java-corba-com-plus_28)
- 25 [www.netbeans.org](http://www.netbeans.org).
- 26 Computer security Principals and practice by WILLIAM STALLINGS & LAWRIE BROWN. S
- 27 [http://en.wikipedia.org/wiki/Digital\\_signature](http://en.wikipedia.org/wiki/Digital_signature)
- 28 JADE TUTORIALJADE PROGRAMMING FOR BEGINNERS, Giovanni Caire (TILAB, formerly CSELT)
- 29 [http://en.wikipedia.org/wiki/Public\\_key\\_certificate](http://en.wikipedia.org/wiki/Public_key_certificate)
- 30 Distributed Systems Principles and Paradigms by Andrew S.tanenbaum, and Maarten Van Steen.
- 31 [http://www.itswtech.org/Lec/Manal%20system%20programming%29/sime\\_ners\\_B/Mobile\\_Agent.pdf](http://www.itswtech.org/Lec/Manal%20system%20programming%29/sime_ners_B/Mobile_Agent.pdf)