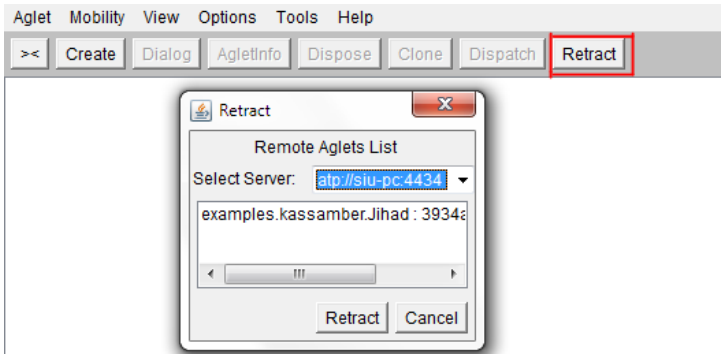
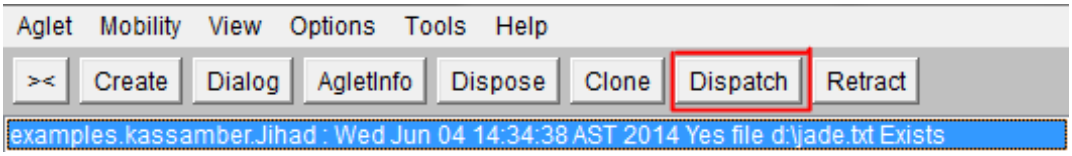
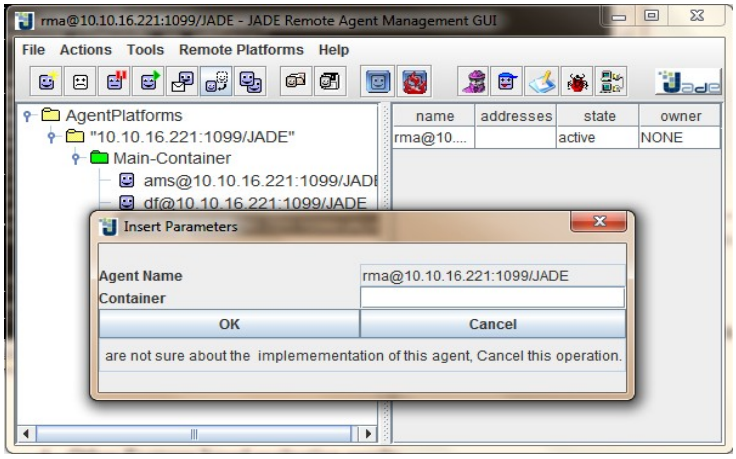
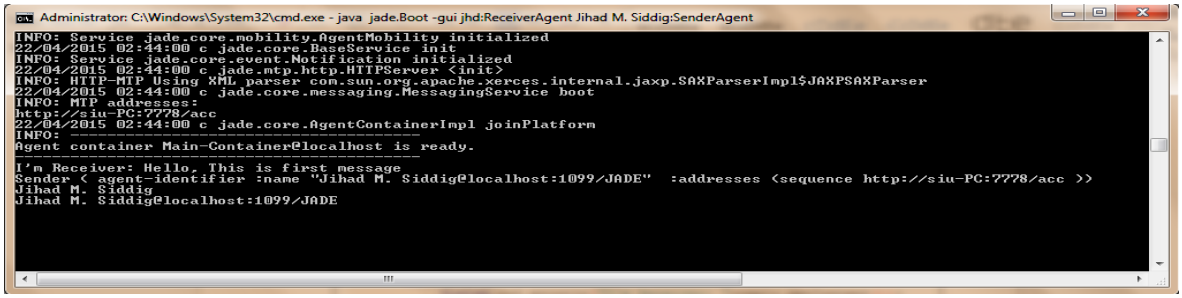



```
public class DispatchingAgent extends Aglet
{
  private Vector curItinVect;
  private URL destination;
  String strDest;
  public void onCreation(Object itin)
  {
       System.out.println("Hi! I have been created..");
       /* INSERT YOUR SERVER & PORT NO: e.g: atp://electra.csse.monash.edu.au:9000 */
       strDest = new String("atp://siu-pc:9000");
       try {
                  destination = new URL(strDest);
       }
       catch (MalformedURLException malURL1)
       {

            System.out.println("mal URL...");
       }
  }
  public void run()
  {

       try {
             dispatch(destination);
             System.out.println("I'm in "+ destination);
       }

       catch (Exception e){
            System.out.println(e.getMessage());
       }


  }
```



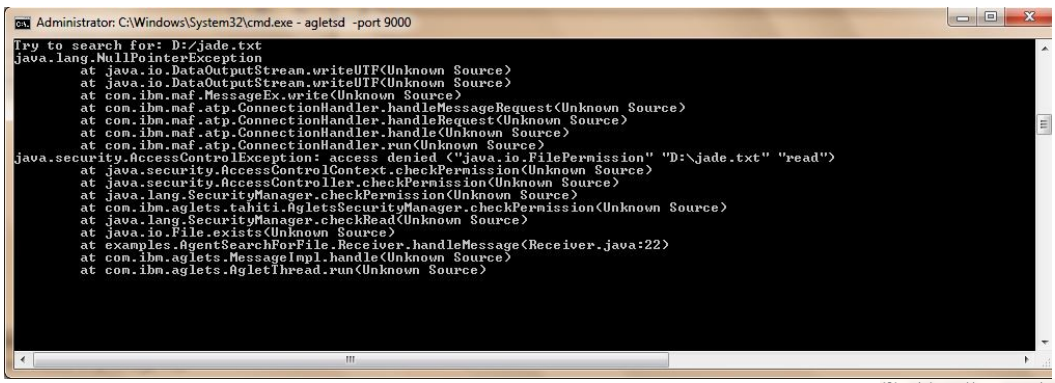

```
public void action()
{
    //System.out.println ("wait for message !");
    ACLMessage reply = myAgent.receive();
    if(reply != null){
        if(reply.getPerformative() == ACLMessage.INFORM){
            System.out.println("I'm Receiver: "+reply.getContent());
            System.out.println("Sender "+reply.getSender());
            System.out.println(reply.getSender().getName());
```

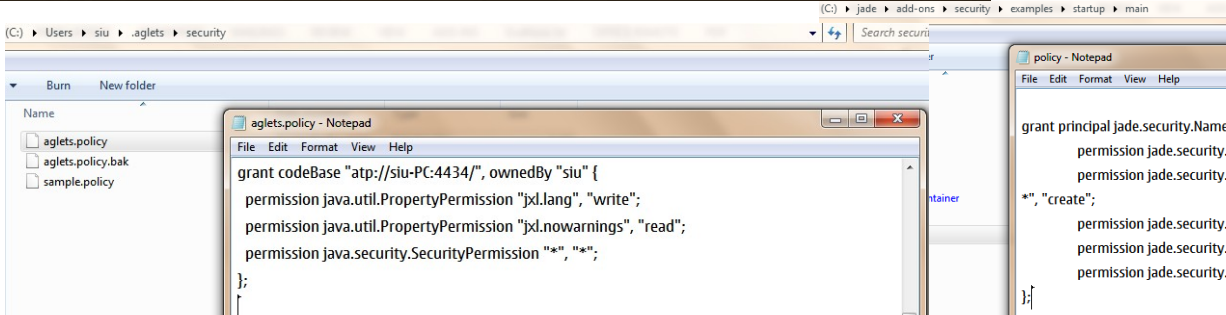
```
public void sendMes
    if (encrypted)
        Iterator it
        while (it.h
            AID rec
            if (myS
                ...
            }
        }

        mySecurityH
    }
    send(msg);
```

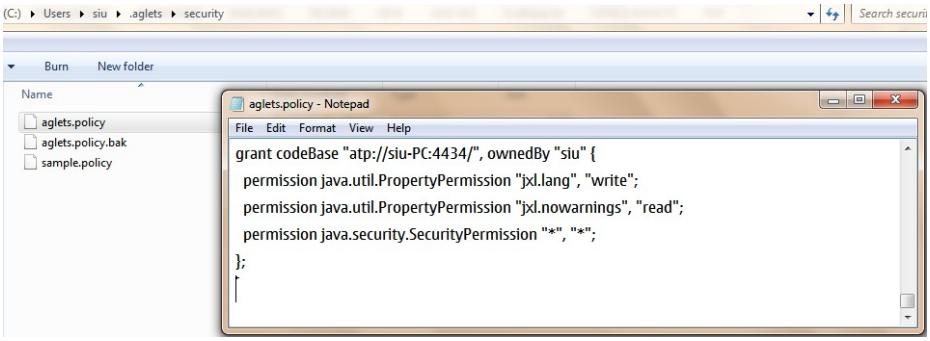
```
public class MAgent extends Agent{

    ContainerID destination = new ContainerID();

    @Override
    protected void setup() {

        super.setup();
        //
        doWait(5000);
        System.out.println(getLocalName()+" has been Started.");
        doWait(5000);
        //
        addBehaviour(new OneShotBehaviour() {
            @Override
            public void action() {
                destination.setName("Container-1");
                myAgent.doMove(destination);
            }
        });
        //
    }
    //
```

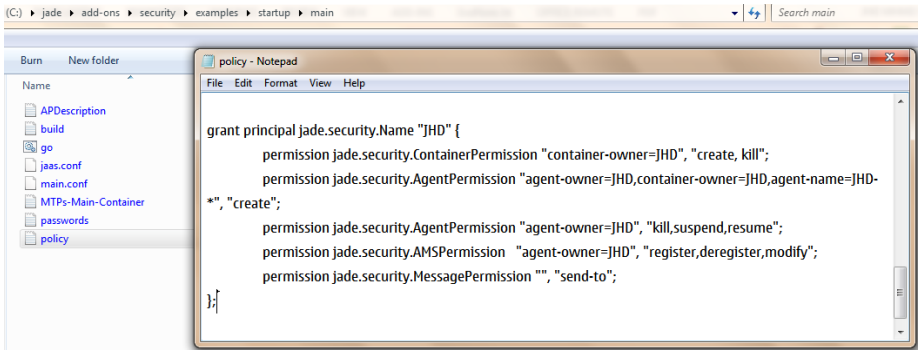
```
public class MAgent extends Agent{

    ContainerID destination = new ContainerID();

    @Override
    protected void setup() {

        super.setup();
        //
        doWait(5000);
        System.out.println(getLocalName()+" has been Started.");
        doWait(5000);
        //
        addBehaviour(new OneShotBehaviour() {
            @Override
            public void action() {
               destination.setName("Container-1");
               myAgent.doMove(destination);
            }
        });
        //
     }
    //
```

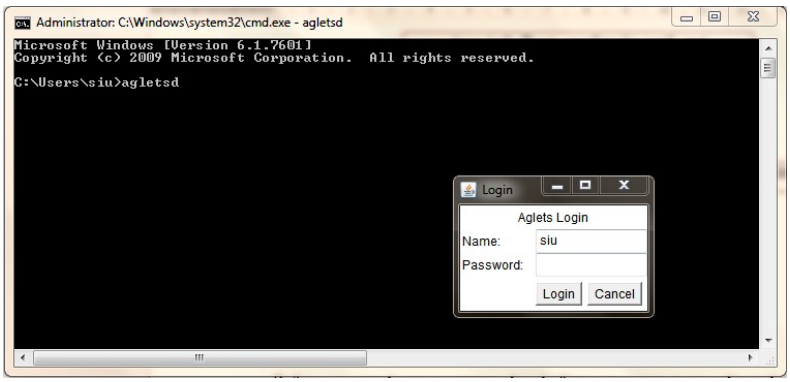
```
public void sendMessage(ACLMessage msg, boolean signed, boolean encrypted) {
    if (encrypted) {
        Iterator it = msg.getAllReceiver();
        while (it.hasNext()) {
            AID receiver = (AID) it.next();
            if (mySecurityHelper.getTrustedPrincipal(receiver.getName()) == null) {
                ...
            }
        }

        mySecurityHelper.setUseEncryption(msg);
    }
    send(msg);
}
public void sendMessage(ACLMessage msg, boolean signed, boolean encrypted) {
    if (signed) {
        mySecurityHelper.setUseSignature(msg);
    }
```

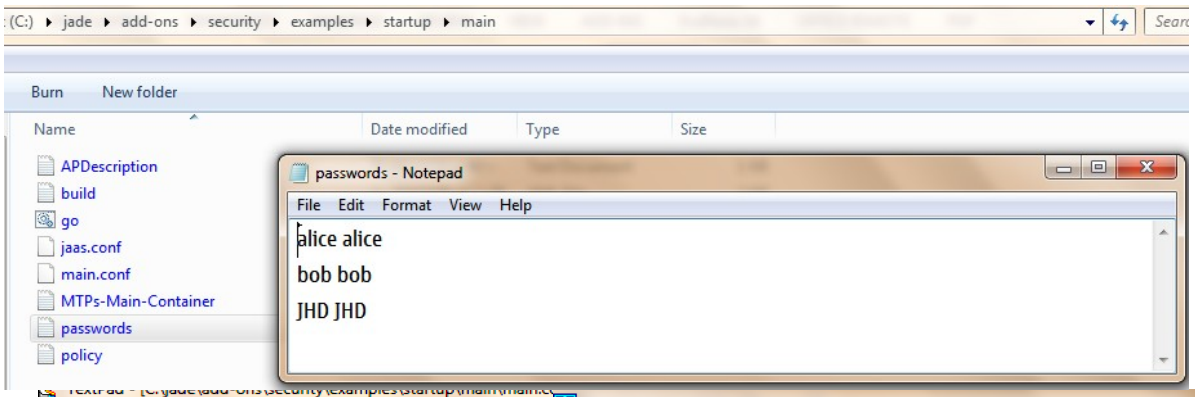


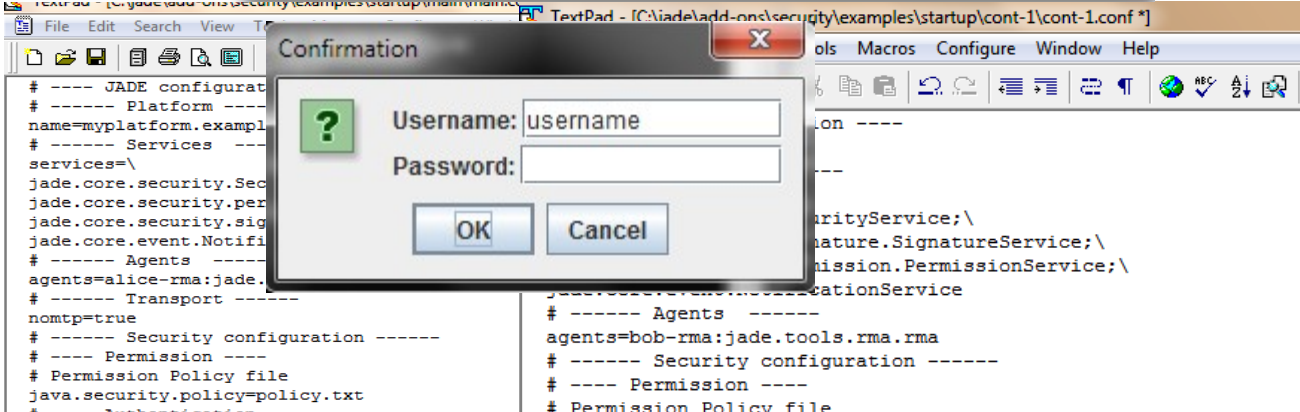



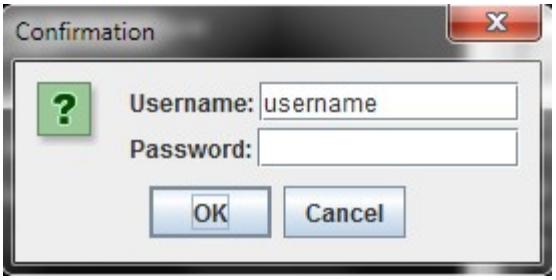

(C:) ▸ Users ▸ siu ▸ .aglets ▸ security
Burn
New folder
Name
aglets.policy
aglets.policy.bak
sample.policy
aglets.policy - Notepad
File Edit Format View Help
grant codeBase "atp://siu-PC:4434/", ownedBy "siu" {
permission java.util.PropertyPermission "jxl.lang", "write";
permission java.util.PropertyPermission "jxl.nowarnings", "read";
permission java.security.SecurityPermission "*", "*";
};


(C:) ▸ jade ▸ add-ons ▸ security ▸ examples ▸ startup ▸ main
Burn
New folder
Name
APDescription
build
go
jaas.conf
main.conf
MTPs-Main-Container
passwords
policy
policy - Notepad
File Edit Format View Help
grant principal jade.security.Name "JHD" {
permission jade.security.ContainerPermission "container-owner=JHD", "create, kill";
permission jade.security.AgentPermission "agent-owner=JHD,container-owner=JHD,agent-name=JHD-*", "create";
permission jade.security.AgentPermission "agent-owner=JHD", "kill,suspend,resume";
permission jade.security.AMSPermission "agent-owner=JHD", "register,deregister,modify";
permission jade.security.MessagePermission "", "send-to";
};


Administrator: C:\Windows\system32\cmd.exe - agletsd
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\siu>agletsd
Login
Aglets Login
Name: siu
Password:
Login
Cancel


(C:) ▸ jade ▸ add-ons ▸ security ▸ examples ▸ startup ▸ main
Burn
New folder
Name
Date modified
Type
Size
APDescription
build
go
jaas.conf
main.conf
MTPs-Main-Container
passwords
policy
passwords - Notepad
File Edit Format View Help
alice alice
bob bob
JHD JHD


File Edit Search View
# ---- JADE configurat
# ------ Platform ----
name=myplatform.exampl
# ------ Services ---
services=\
jade.core.security.Sec
jade.core.security.per
jade.core.security.sig
jade.core.event.Notifi
# ------ Agents -----
agents=alice-rma:jade.
# ------ Transport ------
nomtp=true
# ------ Security configuration ------
# ---- Permission ----
# Permission Policy file
java.security.policy=policy.txt
Confirmation
Username: username
Password:
OK
Cancel
TextPad - [C:\jade\add-ons\security\examples\startup\cont-1\cont-1.conf *]
ols Macros Configure Window Help
ityService;\
nature.SignatureService;\
ission.PermissionService;\
ationService
# ------ Agents ------
agents=bob-rma:jade.tools.rma.rma
# ------ Security configuration ------
# ---- Permission ----
# Permission Policy file




TextPad - [C:\jade\add-ons\security\examples\startup\cont-1\cont-1.conf *]

File Edit Search View Tools Macros Configure Window Help

```
# ---- JADE configuration ----
container=true
# ------ Services  ------
services=\
jade.core.security.SecurityService;\
jade.core.security.signature.SignatureService;\
jade.core.security.permission.PermissionService;\
jade.core.event.NotificationService
# ------ Agents  ------
agents=bob-rma:jade.tools.rma.rma
# ------ Security configuration ------
# ---- Permission ----
# Permission Policy file
java.security.policy=policy.txt
# ---- Authentication ----
# - Type of Prompt -     can be: {Cmdline, Text, Dialog} ('Text' does not work well with 'ant')
jade.security.authentication.logincallback=Dialog
```

TextPad - [C:\jade\add-ons\security\examples\startup\main\main.conf *]

File Edit Search View Tools Macros Configure Window Help

```
# ---- JADE configuration ----
# ------ Platform ------
name=myplatform.example.org
# ------ Services  ------
services=\
jade.core.security.SecurityService;\
jade.core.security.permission.PermissionService;\
jade.core.security.signature.SignatureService;\
jade.core.event.NotificationService
# ------ Agents  ------
agents=alice-rma:jade.tools.rma.rma
# ------ Transport ------
nomtp=true
# ------ Security configuration ------
# ---- Permission ----
# Permission Policy file
java.security.policy=policy.txt
# ---- Authentication ----
# - Type of Prompt -     can be: {Cmdline, Text, Dialog} ('Text' does not work well with 'ant')
jade.security.authentication.logincallback=Cmdline
# - if Cmdline, use this user/pass -
owner=alice:alice
# - Auth module -      can be: {Simple, Unix, NT, Kerberos}
jade.security.authentication.loginmodule=Simple
# - if Simple, use this password file
jade.security.authentication.loginsimplecredfile=passwords.txt
# - JAAS configuration file -
java.security.auth.login.config=jaas.conf
# - Kerberos config parameters -
#java.security.krb5.realm=MYREAL.DOMAIN.COM
#java.security.krb5.kdc=10.1.2.3
#java.security.krb5.conf=/etc/krb5.conf
# ---- end JADE configuration ----
```

```
public boolean handleMessage (Message m){
    int Camera = 50;
    int CameraPrice = 120;
    if (m.sameKind("Camera")){
        System.out.println("A: ask to buy a "+m.getKind());
        m.sendReply("available:"+Camera);
        return true;
    }
    if (m.sameKind("What is camera price")){
        System.out.println("A: "+m.getKind());
        m.sendReply(CameraPrice);
        return true;
    }
    if (m.sameKind("Buying")){
        System.out.println("A: "+m.getKind());
        m.sendReply("How much quantity do you need?");
        return true;
    }
    return false;
```

```
// Add a TickerBehaviour that schedules a request to seller agents every 7 minute
addBehaviour(new TickerBehaviour(this, 7000) {
    protected void onTick() {
        // Update the list of seller agents
        DFAgentDescription template = new DFAgentDescription();
        ServiceDescription sd = new ServiceDescription();
        sd.setType("selling");
        template.addServices(sd);
        try {
            DFAgentDescription[] result = DFService.search(myAgent, template);
            System.out.println("The list of the seller agents:");
            sellerAgents = new AID[result.length];
            for (int i = 0; i < result.length; ++i) {
                sellerAgents[i] = result[i].getName();
                System.out.println(sellerAgents[i].getName());
            }
        }
        catch (FIPAException fe) {
        }

    //Seller agent registrate his name and type of service in DF.
    DFAgentDescription dfd = new DFAgentDescription();
    dfd.setName(getAID());
    ServiceDescription sd = new ServiceDescription();
    sd.setType("selling");
    sd.setName("ebtihal-seller");
    dfd.addServices(sd);
    try {
        DFService.register(this, dfd);
    }
    catch (FIPAException fe) {
        fe.printStackTrace();
    }
```

```
// Receive all proposals/refusals from seller agents
ACLMessage reply = myAgent.receive(mt);
     if (reply != null) {
    // Reply received
    if (reply.getPerformative() == ACLMessage.PROPOSE )
        {
            System.out.println(reply.getSender());
            System.out.println(reply.getContent());
        }
```

$\frac{11.6*100}{13.2}\frac{11.6*100}{13.2}$

```
// Send the cfp to all sellers
ACLMessage cfp = new ACLMessage(ACLMessage.CFP);
for (int i = 0; i < sellerAgents.length; ++i) {

    cfp.addReceiver(sellerAgents[i]);
}
    cfp.setContent(ItemTitle+":"+NumberOfItem);
    cfp.setConversationId("Electronic-trade");
    cfp.setLanguage("English Language");
    cfp.setOntology("Market Ontology");
    cfp.setReplyWith("cfp"+System.currentTimeMillis());
    myAgent.send(cfp);
```

$\frac{7.3*100}{13.2}\frac{7.3*100}{13.2}P_t = \frac{F_t * 100}{F_a}$

$$JADE\ (Ft) = (16 * 10\%) + (12 * 10\%) + (11 * 20\%) + (14 * 30\%) + (8 * 30\%)$$

$$Aglets\ (Ft) = (12 * 10\%) + (10 * 10\%) + (9 * 20\%) + (11 * 30\%) + (0 * 30\%)$$

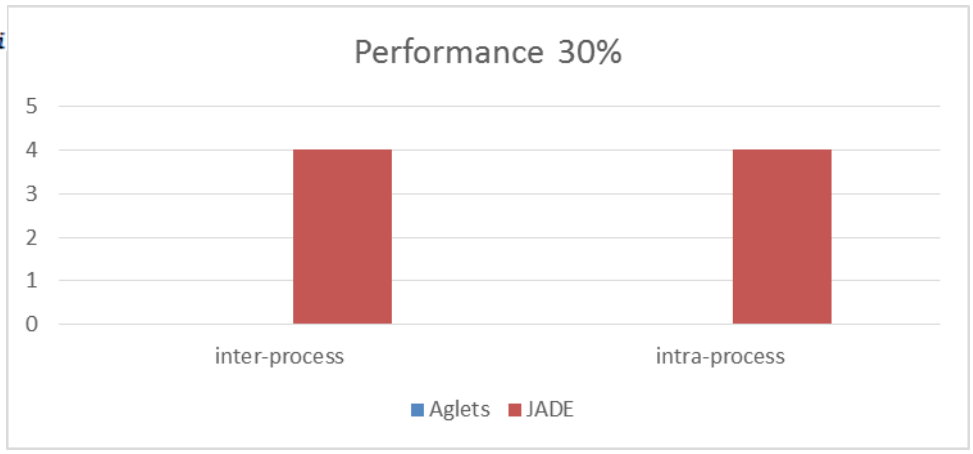
$$F_t = \sum_{i=1}^{n} w_i c_i \quad (JADE)\ ci = 2.4(JADE)\ ci = 2.4(Aglets)\ ci = 0performance\ for:$$

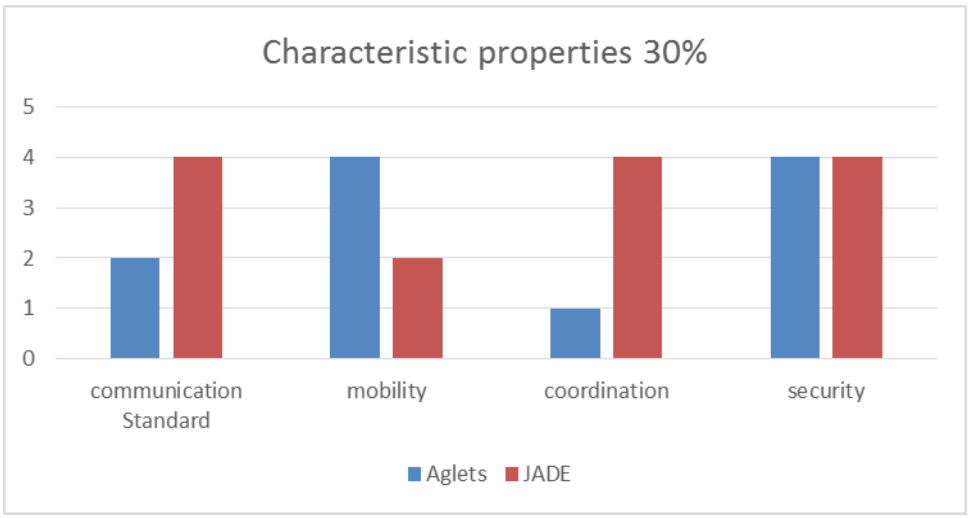
$$(JADE)\ ci = 4.2(Aglets)\ ci = 3.3charateristic\ properties\ for:(JADE)\ ci = 2.2$$

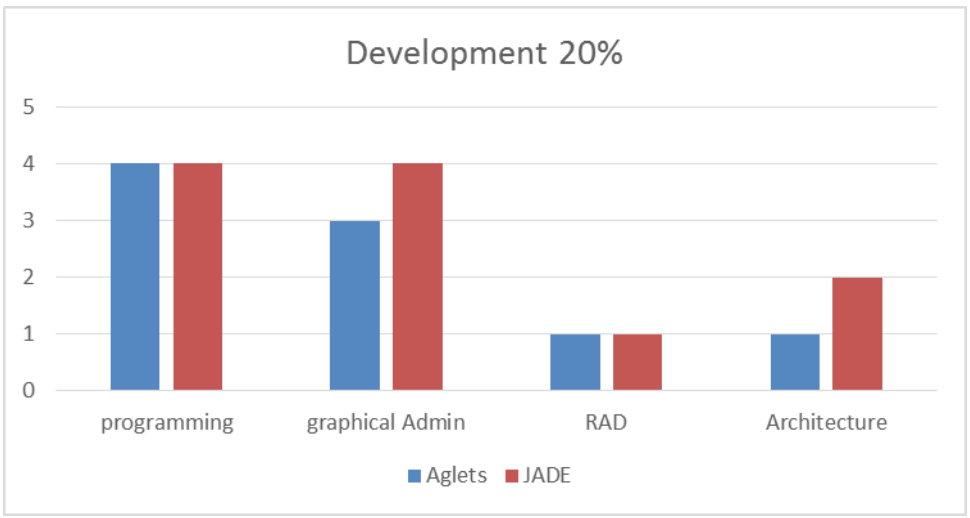
$$(Aglets)\ ci = 1.8development\ for:(JADE)\ ci = 1.2(Aglets)ci = 1.0$$

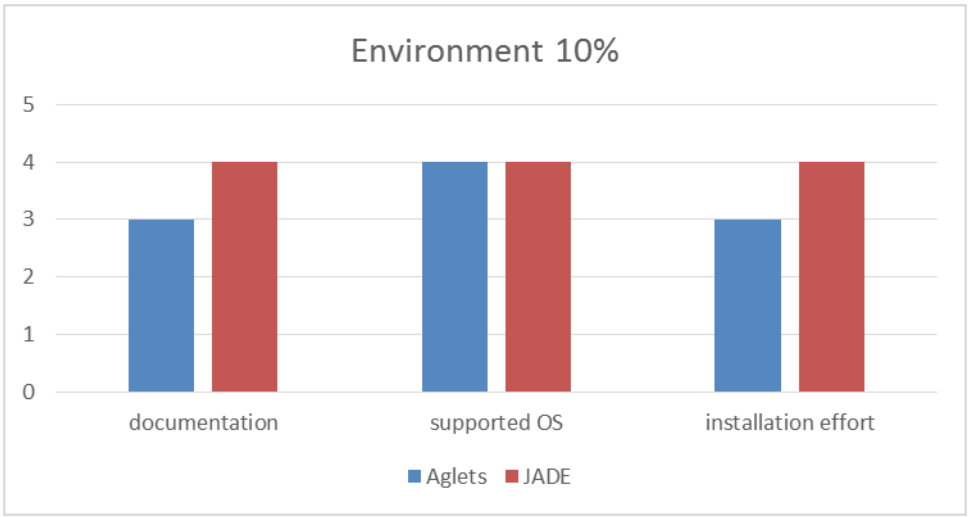
$$environment\ for:(JADE)\ ci = 1.6(Aglets)\ ci = 1.2availability\ for:\ c_i = \sum_k v_k c_i$$

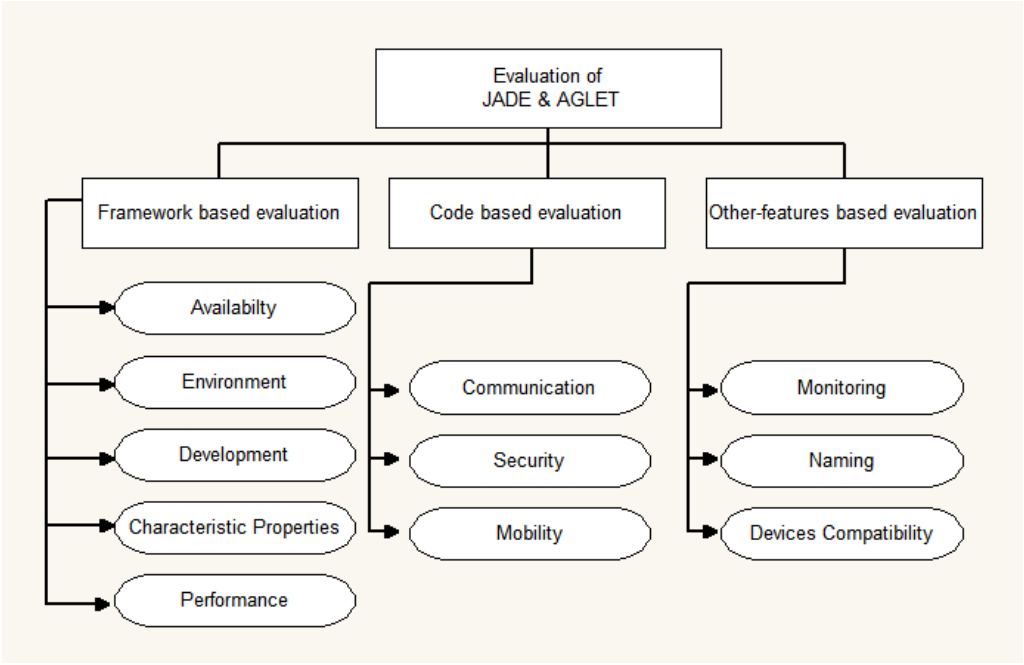
$c_i c_i c_i$



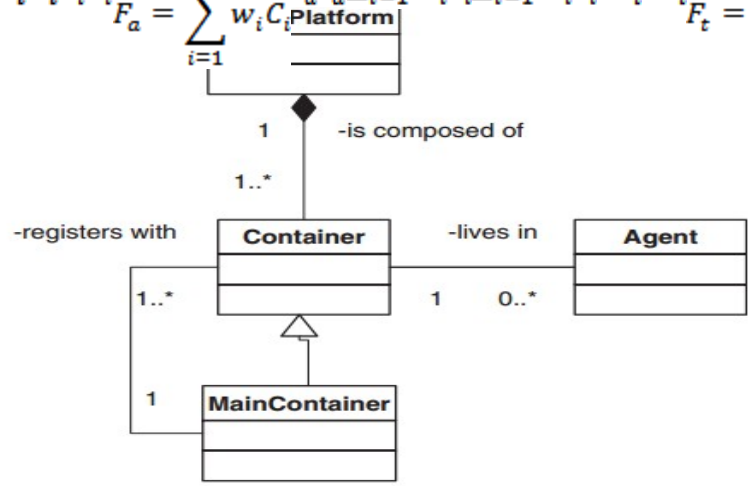

Characteristic properties 30%
5
4
3
2
1
0
communication Standard
mobility
coordination
security
Aglets
JADE


Development 20%
5
4
3
2
1
0
programming
graphical Admin
RAD
Architecture
Aglets
JADE

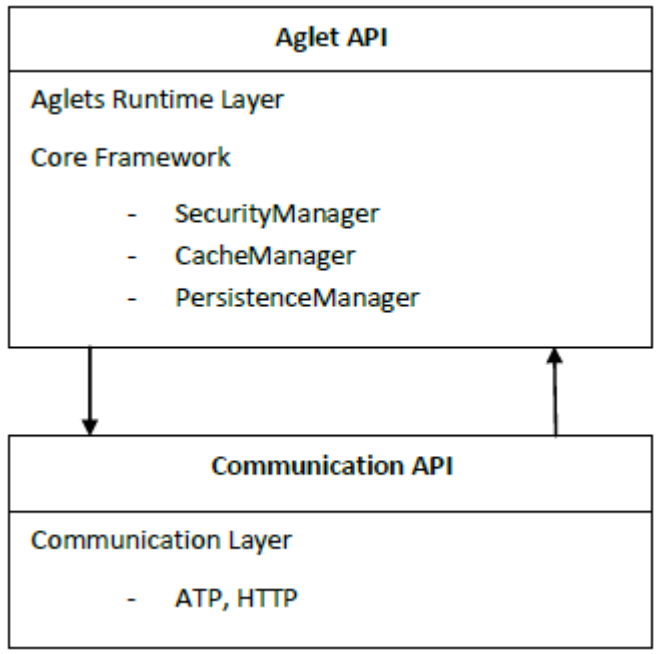

Environment 10%
5
4
3
2
1
0
documentation
supported OS
installation effort
Aglets
JADE

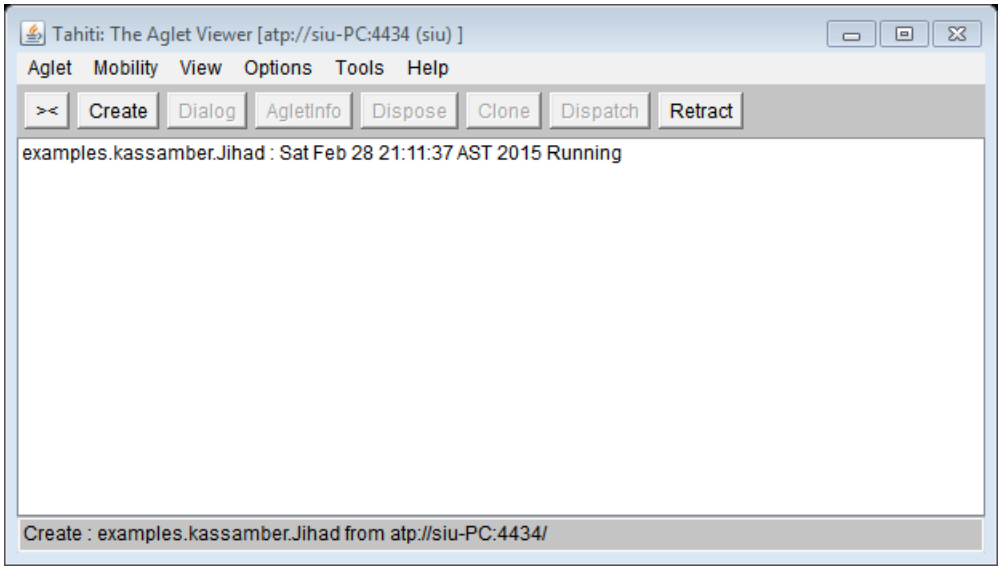




$$P_t = \frac{F_t * 100}{F_a} \quad \boldsymbol{P_t P_t} \quad C_i C_i \quad F_a = \sum_{i=1}^{n} w_i C_i \quad F_a F_a \sum_{i=1}^{n} w_i c_i \sum_{i=1}^{n} w_i c_i \; w_i \; w_i \quad F_t = \sum_{i=1}^{n} w_i c_i \quad F_t F_t$$

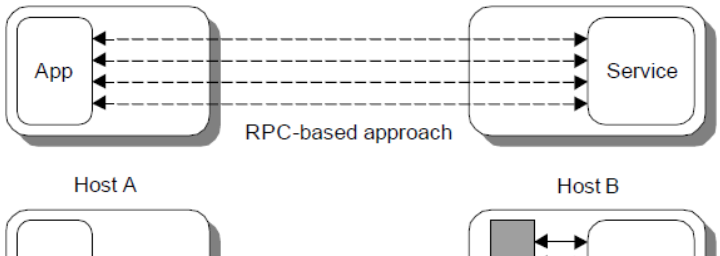


$$c_i = \sum_k v_k v_k v_k$$

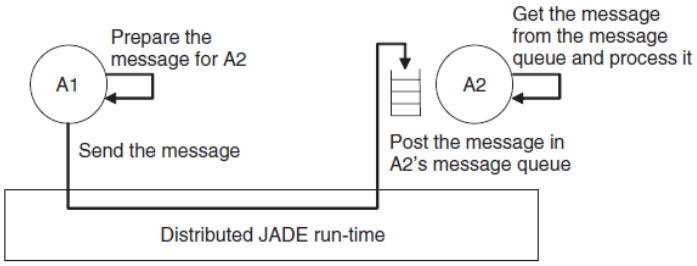


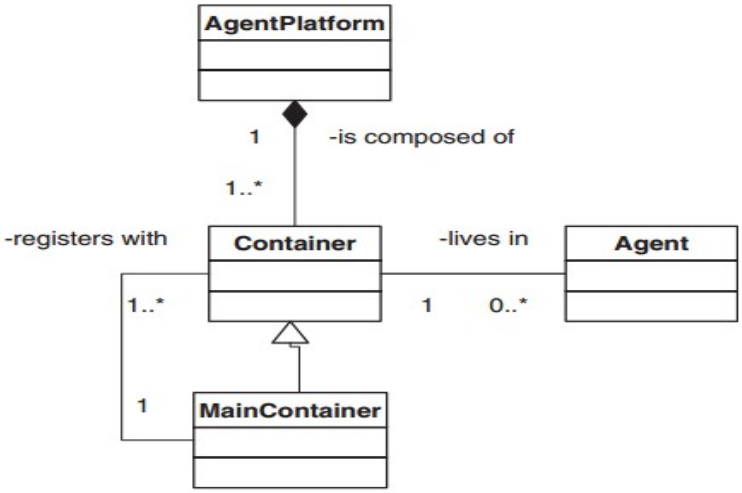



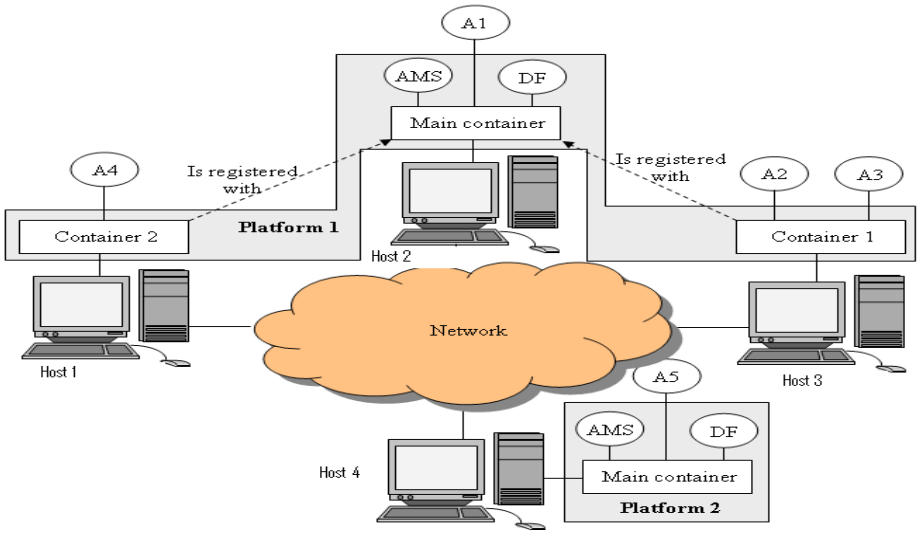

```
// Receive all proposals/refusals from seller agents
ACLMessage reply = myAgent.receive(mt);
     if (reply != null) {
    // Reply received
    if (reply.getPerformative() == ACLMessage.PROPOSE )
        {
            System.out.println(reply.getSender());
            System.out.println(reply.getContent());
        }
```

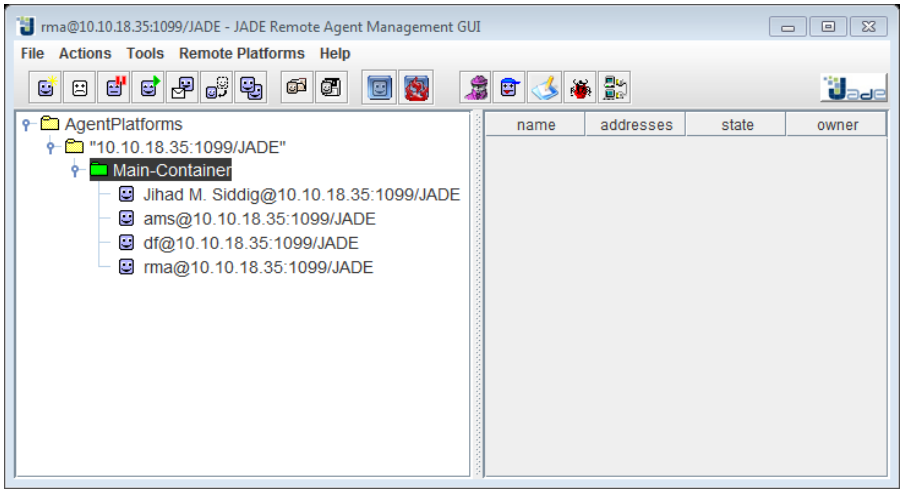


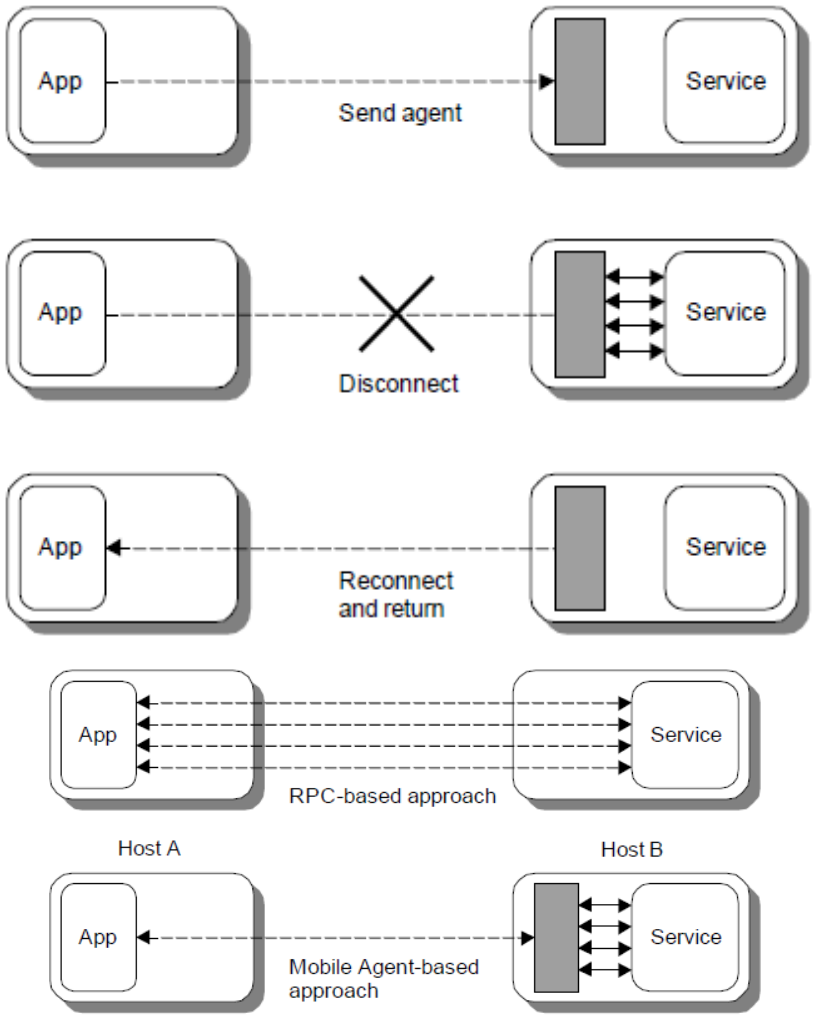
```
// Send the cfp to all sellers
ACLMessage cfp = new ACLMessage(ACLMessage.CFP);
    cfp.addReceiver(sellerAgents[i]);
    cfp.setContent("Hello I'm agent");
    myAgent.send(cfp);
```




rma@10.10.18.35:1099/JADE - JADE Remote Agent Management GUI
File Actions Tools Remote Platforms Help
AgentPlatforms
"10.10.18.35:1099/JADE"
Main-Container
Jihad M. Siddig@10.10.18.35:1099/JADE
ams@10.10.18.35:1099/JADE
df@10.10.18.35:1099/JADE
rma@10.10.18.35:1099/JADE
name
addresses
state
owner


App
Send agent
Service
App
Disconnect
Service
App
Reconnect
and return
Service
App
RPC-based approach
Service
Host A
Host B
App
Mobile Agent-based
approach
Service


بِسْمِ ٱللَّهِ ٱلرَّحْمَـٰنِ ٱلرَّحِيمِ

ٱقْرَأْ بِٱسْمِ رَبِّكَ ٱلَّـذِى خَلَقَ ۝١ خَلَقَ ٱلْإِنـسَـٰنَ مِنْ عَلَقٍ ۝٢
ٱقْرَأْ وَرَبُّكَ ٱلْأَكْرَمُ ۝٣ ٱلَّذِى عَلَّمَ بِٱلْقَلَـمِ ۝٤ عَلَّمَ ٱلْإِنسَـٰنَ مَا
لَـمْ يَعْلَمْ ۝٥

صدق الله العظيم

**بسم الله الرحمن الرحيم**

Sudan University of Science and Technology

College of Graduate Studies

Title:

# Comparison and evaluation of performance of mobile agent toolkits (JADE and Aglet)

# مقارنة وتقويم الاداء لأدوات الوكيل النقال (JADE و Aglet)

*A thesis submitted for partial fulfillment for the requirements of M.Sc Degree in Computer science*

**By: Jihad Mohammed Siddig Ahmed**

**Supervised by: Dr. Ali Ahmed Al-Faki**

**2015**

**Verse**

## Dedication

**To My:**

***Parents***

**…**

***Sisters and Brothers***

**…**

***Teachers***

**…**

***Friends***

# Acknowledgement

I would like to grasp this opportunity to express of my deepest gratitude to all who have helped me directly and indirectly towards the successful completion of this research.

Foremost, I would like to express of my sincere gratitude to my supervisor **Dr. Ali Ahmed Al-Faki** Assistant Professor in Karary University in Sudan, he was agreed to supervise of this research in a late time although he is busier. Also for his advice, constant, support and valuable suggestions throughout the course of this research.

Besides my supervisor, I would like to thank **Dr. Amir Abdelfattah Ahmed Eisa,** Associated Professor in Sudan University of Science and Technology in Sudan, for his helped us to understand multi-agent system. Especially for me to choose this research topic. Also, I am grateful for entire teaching staff of SUST College of computer science and information technology for their help during my M.Sc.

Last but not least, I am thankful to my parent, sisters, brothers and colleagues for their support and encouragement to pursue my interests.

The thanks firstly and lastly for Allah.

**Jihad Mohammed Siddig**

## Abstract

Mobile agent applications becomes very spread in the recent years thus for its importance. This importance allows many of the toolkits to use in the development mobile agent applications; each toolkit has its own advantages and disadvantage. Therefore, we find that the software developers may hesitate of choosing the appropriate toolkit. Although many of comparison studies in this area have been submitted, but the continuous update make the developers need to an up-to-date evaluation on the existing toolkits. In this research, we compare between JADE (Java Agent DEvelopment framework) and Aglet based on three phases of evaluation. Phase one is frame based evaluation (we apply the framework of evaluate the agent toolkits proposed by Elhadi Shakshuki), phase two of evaluation based on running some scenarios and settings on each toolkit, while the third phase is the evaluation based on the features included in each toolkit which facilitate of using the toolkit and managing of mobile agent. The results obtained based on these phases showed the preference of JADE versus Aglet. This preference is due to the continuous update on JADE (The last version of JADE - JADE 4.3.3 - released on 11/12.2014 and previous version - JADE 4.3.1- released on 06/12/2013) while the last version of Aglet - Aglet 2.0.2 - released before 2010.

## المستخلص

تطبيقات الوكيل النقال أصبحت منتشرة بصورة واسعة في السنوات الأخيرة ولذلك لما اكتسبته من أهمية، هذه الأهمية جعلت العديد من الأدوات تستخدم في تطوير تطبيقات الوكيل النقال، وكل أداة لها ايجابياتها وسلبياتها، لذلك نجد أن مطوري البرامج قد يترددون في اختيار الأداة المناسبة. وبالرغم من أن العديد من الدراسات في هذا المجال قد تم تقديمها ولكن التحديث المستمر على هذه الأدوات يجعل مطوري البرامج بحاجة إلى مقارنة حديثة بين الأدوات الموجودة. في هذا البحث تمت المقارنة بين JADE و Aglet بناءاً على ثلاث مراحل للتقييم. المرحلة الأولى: تعتمد على استخدام منهجية للتقييم (تم تطبيق المنهجية المقترحة بواسطة الهادي شكشوكي). المرحلة الثانية: تعتمد على تنفيذ بعض السيناريوهات والإعدادات أما المرحلة الثالثة تعتمد على التقييم إعتماداً على مقارنة الميزات الموجودة في كل أداة والتي تسهل من استخدام الأداة وإدارة الوكلاء. النتائج التي تم الحصول عليها أظهرت أفضلية JADE على Aglet، وتُعزي هذه الأفضلية إلى التحديث المستمر على JADE (حيث أن أخر نسخة من JADE (JADE 4.3.3 تم إطلاقها في 11/12/2014 والنسخة السابقة منها (JADE 4.3.1 تم إطلاقها في 6/12/2013 بينما نجد أن أخر نسخة من Aglet (Aglet 2.0.2 تم إطلاقها قبل العام 2010.

# Table of Contents

# List of Tables

**Acronyms**

| Acronym | Meaning |
|---|---|
| **ACL** | Agent Communication Language |
| **AGLET** | Agent + applet |
| **AID** | Agent Identifier |
| **AMS** | Agent Management System |
| **API** | Application Programming Interface |
| **ASDK** | Aglet Software development Kit |
| **ATP** | Agent Transfer Protocol |
| **CFP** | Call For Proposal |
| **CMD** | Command prompt |
| **CT** | Container Table |
| **DF** | Directory Facilitator |
| **FAQ** | Frequently Asked Questions |
| **FIPA** | Foundation for Intelligent Physical Agent |
| **GADT** | Global Agent Descriptor Table |
| **GUI** | Graphical User Interface |
| **IPMS** | Inter Platform Mobility Service |
| **JADE** | Java Agent DEvelopment Framework |
| **KIF** | Knowledge Interchange Format |
| **KQML** | Knowledge Query Manipulation Language |
| **LGPL** | Library Gun Public License |
| **MAS** | Multi Agent System |
| **MASIF** | Mobile Agent System Interoperability Facility |
| **MTS** | Message Transport Service |
| **OS** | Operating System |
| **RAD** | Rapid Application Development |
| **RMI** | Remote Method Invocation |

Abstract

# CHAPTER ONE

## Introduction

# Chapter (1)
# Introduction

## 1.1 **Research Background**

Mobile agent is becoming new paradigm of software engineering[1]. So that many organizations has started to use the mobile agent technology especially in Networks, distributed data and the internet environments[2]. There exist a huge number of approaches, toolkits, and platforms of different quality and maturity[1]. With this interest there are many of software platforms have been developed, and many of research has been done on developing mobile agent while other research has been done on the platform that use to develop mobile agent, some of this platforms still in use while other has not been used.

### 1.1.1 Agent and Multi-agent system

"An agent is a computer system that is situated in some environment and capable of autonomous action in this environment in order to meet its design objectives[3]". Any control system can be viewed as an agent, for example the Thermostat: Thermostat goal is to maintain the room temperature, has a sensor to perceive it's environment and has two action "*HeatOn*" execute if the room is too cold and "*HeatOff*" execute if the temperature is Ok. The agent has properties such as Autonomy - which distinguish the agent-based system from the traditional software systems– and Reactivity, Proactiveness and Social ability –which are give the agent the property of intelligence -. Intelligent agents are a new paradigm for developing software applications agent-based system has been hail as: “the new revolution in software”[4]. Moreover, Multi-agent system "is one that consists of a number of agents, which interact with one another, typically by exchanging messages through some computer network infrastructure. In the most general case, the agents in a multi-agent system will be representing or acting on behalf of users or owners with very different goals and motivations. In order to successfully interact. These agents will thus require the ability to cooperate, coordinate, and negotiate with each other, in much the same way that we cooperate, coordinate, and negotiate with other people in our everyday lives[3].

### 1.1.2 Mobile agent

"A mobile agent is a composition of computer software and data which is able to migrate (move) from one computer to another autonomously and continue its execution on the destination computer[5]".

### 1.1.3 Why Using Mobile agents

**"They reduce the network load.** Distributed systems often rely on communications protocols that involve multiple interactions to accomplish a given task. This is especially true when security measures are enabled. The result is a lot of network traffic. Mobile agents allow you to package a conversation and dispatch it to a destination host where the interactions can take place locally, see Figure 1.1. Mobile agents are also useful when it comes to reducing the flow of raw data in the network[6]".

Figure 1.: Mobile agent and network reduction

**"They overcoming network latency.** Critical real-time systems need to respond to changes in their environments in real time. Controlling such systems through a factory network of a substantial size involves significant latencies. For critical real-time systems, such latencies are not acceptable. Mobile agents offer a solution, since they can be dispatched from a central controller to act locally and directly execute the controller's directions[6]".

**"They encapsulate protocols.** When data are exchanged in a distributed system, each host owns the code that implements the protocols needed to properly code outgoing data and interpret incoming data, respectively. However, as protocols evolve to accommodate new efficiency or security requirements, it is a cumbersome if not impossible task to upgrade protocol code properly. The result is often that protocols become a legacy problem. Mobile agents, on the other hand, are able to move to remote hosts in order to establish "channels" based on proprietary protocols[6]".

**"They execute asynchronously and autonomously.** Often mobile devices have to rely on expensive or fragile network connections. That is, tasks that require a

continuously open connection between a mobile device and a fixed network will most likely not be economically or technically feasible. Tasks can be embedded into mobile agents, which can then be dispatched into the network. After being dispatched, the mobile agents become independent of the creating process and can operate asynchronously and autonomously. The mobile device can reconnect at some later time to collect the agent[6]".

Figure 1.: Mobile agent and disconnected operation

"**They adapt dynamically.** Mobile agents have the ability to sense their execution environment and react autonomously to changes. Multiple mobile agents possess the unique ability to distribute themselves among the hosts in the network in such a way as to maintain the optimal configuration for solving a particular problem[6]".

"**They are naturally heterogeneous.** Network computing is fundamentally heterogeneous, often from both hardware and software perspectives. As mobile agents are generally computer- and transport-layer-independent, and dependent only on their execution environment, they provide optimal conditions for seamless system integration[6]".

"**They are robust and fault-tolerant.** The ability of mobile agents to react dynamically to unfavorable situations and events makes it easier to build robust and fault-tolerant distributed systems. If a host is being shut down, all agents executing on that machine will be warned and given time to dispatch and continue their operation on another host in the network[6]".

### 1.1.4 Mobile agent applications

- Electronic commerce.
- Personal assistance.
- Secure brokering.
- Distributed information retrieval.
- Telecommunication networks services.
- Workflow applications and groupware.

- Monitoring and notification.
- Information dissemination.
- Parallel processing.

### 1.1.5 Toolkits for Mobile agent Systems

Java has generated a flood of experimental mobile agent systems. Numerous systems are currently under development, and most of them are available for evaluation on the Web[6].

- **Aglets**.
- Anchor
- Zeus
- Odyssey.
- Concordia.
- Voyager.
- Agent Tcl.
- Ara
- TACOMA.
- ….
- **JADE**.

This thesis is concern with compare between **JADE** which is a software platform that provides basic middleware-layer functionalities which are independent of the specific application and which simplify the realization of distributed applications that exploit the software agent abstraction[3]. And **Aglets**: which is a Java based mobile agent toolkit developed by IBM. The goal was to bring the flavor of mobility to the applet. The term aglet is indeed a hybrid word from agent and applet.

## 1.2 **Research questions**

From the above introduction a main question can be dawn on to developers, what toolkit may be suitable to develop mobile agent?

This question can divide into two sub questions:

1. Which criteria should be consider when we want to developing mobile agent system?
2. What is the features of each tool?

## 1.3 **Problem Statement**

The enthusiastic interest in mobile agent technology make several platforms has been developed. Some of them used for research purposes while others as commercial products[7, 8]. For that, a common problem is decision of which platform that developers can use[2, 9]. It is difficult for developers to select the appropriate toolkit [10, 11]. How to choose suitable platform when we want to develop mobile agent and which criteria should consider.

## 1.4 **Research Scope**

This research focus on the comparison between **JADE 4.3.1** and **Aglet 2.0.2** when they use to develop mobile agent systems.

## 1.5 **Research Objectives**

The objectives of this research is to:

1. Make a comparison between **JADE 4.3.1** and **Aglet 2.0.2** based on three phases for comparison.
2. Apply the framework of agent toolkits evaluation proposed by Elhadi Shakshuki[10].
3. Help developers to choose suitable toolkit for developing mobile agent.

## 1.6 **Research Motivation**

The importance of this research is lie in the existence of many mobile agent software tools. Some of them has been used in the past[8], but has not still working either for new concepts of agent based system or it has not complies with international agent standard -like FIPA standard. While some of other toolkits are still working. Developers when they need to build agent based system needs to an **up to date** evaluation on the existed toolkits.

## 1.7 **Research Structure**

This study is broadly organized into five chapters, in chapter (1) we gave a brief introduction to agent system, and its importance also the common tools for mobile

agent applications was introduced. Chapter (2) has divided into six sections, in section one, and two give a brief glance about the subject of comparison toolkits, section three summarizes some information about the organization of agent application and agent communication paradigms, section four include the explanation of the framework used in the comparison, section five show the related work done by others whereas section six include the researcher point of view about the section five. Chapter (3) contains the explanation of the comparison phases. Chapter (4) show the analysis and implementation of applying the phases of comparison. Chapter (5) contains a brief conclusions and recommendations for the future work.

## 1.8 **chapter conclusion**

this chapter was classified into seven paragraphs, paragraph one research background which contain a brief introduction on agents and multi agent systems, the advantages of using mobile agents, the application of mobile agents and the toolkits used to develop mobile agents , while other paragraphs contain the questions, problem statement, scope, objective, motivation and research organization

# CHAPTER TWO

## Background and Literature Review

# Chapter (2)

# Literature Review

## 2.1 (Java Agent DEvelopment Framework (JADE).

In this section we will give a brief description and overview about the JADE toolkit, the overview include basic information of the language of developing agent, the standard was follow, the company that holding the copyright of JADE. In addition, we will give a brief introduction about the architecture, communication and mobility paradigm (including inter-platform and intra-platform mobility) of the JADE.

### 2.1.1 JADE Overview

JADE (Java Agent DEvelopment Framework) is a software Framework fully implemented in the Java language. It simplifies the implementation of multi-agent systems through a middle-ware that complies with the FIPA (Foundation for Intelligent Physical Agent) specifications and through a set of graphical tools that support the debugging and deployment phases. A JADE-based system can be distributed across machines (which not even need to share the same OS) and the configuration can be controlled via a remote GUI. JADE is completely implemented in Java language and the minimal system requirement is the version 5 of JAVA. Besides the agent abstraction, JADE provides a simple yet powerful task execution and composition model, peer to peer agent communication based on the asynchronous message passing paradigm, a yellow pages service supporting publish subscribe discovery mechanism and many other advanced features that facilitates the development of a distributed system. JADE is free software and is distributed by Telecom Italia, the copyright holder, in open source under the terms and conditions of the LGPL (Lesser General Public License Version 2) license. Besides the JADE Team[12].

Figure 2.1 illustrate the GUI of JADE including some running agents.

Figure 2.: JADE Remote Agent Management GUI

### 2.1.2 JADE Architecture

A JADE platform is composed of agent **containers** that can be distributed over the network. Agents live in containers which are the Java process that provides the JADE run-time and all the services needed for hosting and executing agents. There is a special container, called the **main container**, which represents the bootstrap point of a platform: it is the first container to be launched and all other containers must join to a main container by registering with it. Figure 4.2 shows the main architectural .[elements of a JADE platform[12

Figure 2.: JADE Architecture

:The main container has the following special responsibilities

- Managing the container table (CT), which is the registry of the object references and ;transport addresses of all container nodes composing the platform
- Managing the global agent descriptor table (GADT), which is the registry of all agents ;present in the platform, including their current status and location
- Hosting the AMS and the DF, which are the two special agents that provide the agent .management and white page service

Figure 2.3 show the relationships between the main architectural elements of JADE[12].

Figure 2.: Relationship between the main architectural elements

### **Agent Communication**2.1.3

Agent communication is probably the most fundamental feature of JADE and is implemented in accordance with the FIPA specifications. The communication paradigm is based on asynchronous message passing[12].

Figure 2.: JADE asynchronous message paradigm

The particular format of messages in JADE is compliant with that defined by the FIPA-ACL message structure[12]. Each message includes at least the following fields:

- .Sender
- .Receivers
- contents
- .Performative

*Figure 2.: part of code for CPF message performative*

*Figure 2.: part of code for PROPOSE performative*

### 2.1.4 **Intra-platform mobility**

JADE has a provided platform service called the agent Mobility Service, which implements intra-platform mobility. This provides software agents with the ability to move among different containers within the same platform. This mechanism, however, does not allow agents to move to containers belonging to other platforms[12].

### 2.1.5 **Inter-platform mobility**

The Inter-Platform Mobility Service (IPMS) provide platform-to-platform mobility for JADE agents. The IPMS is specifically designed to be as transparent as possible to the agent programmer by ensuring that inter-platform migration is as straightforward as intra-platform migration[12].

## 2.2 **Aglet Software Development Kit (ASDK).**

In the next sections, we will give a brief overview about the Aglet, language used in developing agent using aglet, the company that holding the copyrights and in the Figure 2.7 will show the main GUI the use for create and managing mobile agent.

In addition, we will give a brief information about the Aglet architecture and the main framework that provide the necessary services for managing agent (including management components and communication layer).

### 2.2.1 **Aglet Overview**

Aglets is a Java mobile agent platform and library that eases the development of agent based applications. An aglet is a Java agent able to autonomously and spontaneously move from one host to another. Originally developed at the IBM Tokyo Research Laboratory. Aglets completely made in Java, Aglets includes both a complete Java mobile agent platform, with a stand-alone server called Tahiti, It provides users with a good GUI and allows users to create & dispatch an agent, monitor it, dispose it off when required. It gives user the ability to set agent's access privileges on the server[13][13][13].

Figure 2.: Tahiti - The GUI of Aglet

and a library that allows developer to build mobile agents and to embed the Aglets technology in their applications[13][13][13]. The term aglet is a word combining agent and applet. Currently, stable release of Aglets are available in the 2.0 series, and 2.0.2 is the latest one.

### 2.2.2 **Aglet Architecture**

The aglets framework consists of three layers. On the top is the application layer with the user-define aglets. These aglets interface with the aglets API and its implementation, the aglets runtime layer, needed for the proper management and execution of aglets consists of two parts: a core framework and a set of extensible system management components[6].

### 2.2.3 **Core Framework**

This framework provides the core services necessary for executing aglets, including basic operations such as creation, cloning, dispatching, retraction, deactivation, and activation. The primary functions of the core framework are to securely manage the following[6]:

- Initialization and serialization / deserialization of aglets.
- Class loading and transfer.
- Aglet references and garbage collection.

### 1.1Management components

The aglet runtime layer also features the following components, which are designed to be customizable. They are intended for the creator of an aglet server and can conveniently be used to optimize the performance and security of given server.

- The persistence manager component.
- The cache manager component.
- The security manager component.

The management components are define as either interfaces or abstract classes and can thus be implemented or extended to fit the specific environment in which a given aglet server is deployed[6].

### 1.2Communication Layer

The aglet runtime has no build-in mechanism for transporting over the network. Instead, the aglet runtime layer interfaces with the generic communication layer[6].

Figure 2.: Aglet Architecture

### 2.3Foundation for Intelligent Physical Agent (FIPA)

FIPA was established in 1996 as an international non-profit association to develop a collection of standards relating to software agent technology. The initial membership, a collection of academic and industrial organizations, drew up a set of statutes guiding the production of a set of de jure standard specifications for software agent technologies. At that, time software agents were already very well known in the academic community but have to date received only limited attention from commercial enterprises beyond an exploratory perspective. The consortium agreed to

produce standards that would form the bedrock of a new industry by being usable across a vast number of applications[12].

At the core of FIPA is the following set of principles:

1. Agent technologies provide a new paradigm to solve old and new problems.
2. Some agent technologies have reached a considerable degree of maturity.
3. To be of use some agent technologies require standardization.
4. Standardization of generic technologies has been shown to be possible and to provide effective results by other standardization for a.
5. The standardization of the internal mechanics of agents themselves is not the primary concern, but rather the infrastructure and language required for open interoperation.

### 2.3.1 **ACL**

It is a language that rely on speech act theory and that provide a separation between the communicative acts and the content language. The primary features of FIPA ACL are the possibility of using different content languages and the management of conversations through predefined interaction protocols[12].

### 2.3.2 **KQML**

The first agent communication language it was developed in the early 1990s as part of the US government's ARPA Knowledge Sharing Effort[12].

### 2.3.3 **KQML KIF.**

It is a language and protocol for exchanging information and knowledge that defines a number of performative verbs and allows message content to be represented in a first-order logic[12].

## 2.4**Comparison Evaluation Based on Framework**

According to methodology proposed by Shakshuki{Shakshuki, 2005 #27}, there are five criteria catalogue for each toolkit, including availability, environment, development, characteristic properties, and performance. Each criteria catalogue, *i*, is assigned a weight, *wi*. Each criterion consists several aspects of toolkit feature. Each aspect of the toolkit feature is assigned an integer value between *0* and *4* in order to have numeric value associated with each toolkit[10]. These values assigned as follow:

- *.there is no hint about this feature–0*
- *.Implies bad-1*
- *.Implies poor-2*
- *.implies good-3*
- *.implies very good-4*

For each criteria catalogue, the value of each aspect of the associated toolkit feature is added using the following equation:

Where, *k* is the number of evaluated aspects for criteria catalogue *i*.
The feature sum of a toolkit *t* is calculated using the following formula:

Where, is the weight assigned to category *i*, and *n* in the number of categories considered for evaluation, and = 100%.
The available points that are considered,, of a toolkit are calculated using the following formula:

Where, is the best value for each criterion.
The **percentage** of toolkit is calculated using the following formula:

## 2.5**Previous Studies on agent-based applications toolkits.**

Large-scale realization of agent applications requires frameworks, methodologies, and tools that support effective development of agent systems. Nowadays many agent development toolkits and platforms of different quality and maturity are available and have been employed for different applications in different parts of the world. There is so far no consensus about which toolkit is best for agent development. Thus, it is worth studying various agent toolkits in depth, and to analyze their strengths & weaknesses[14]. The study compare JADE and Aglet according to criteria of Nature of Product, Standard Implemented, Communication Technique, Security Mechanism, agent Mobility and Migration Mechanism. And summarized their qualitative comparison result as follow:

| ***Toolkit*** <br> ***Feature*** | ***Aglet*** | ***JADE*** |
|---|---|---|
| Nature of product | Free, Open source | Free, Open source |
| Standard Implement | MASIF | FIPA |
| Communication Technique | Synchronous, Asynchronous | Asynchronous |
| Security Mechanism | Poor | Good |
| Agent Mobility | Weak | Not-so-weak |
| Migration Mechanism | Socket | RMI |

*[ Table 2.1: Comparison of JADE and Aglet Toolkits [14*

According to the comparison, JADE agent development toolkit seems most appealing. It is open source platform, purely designed in Java, provides consistency in API, and supports different kinds of devices operating in internet. It provides good security features and supports sound agent mobility. Whereas Aglet also does not comply with FIPA, lacks security and scalability. Thus JADE agent development toolkit is most balanced toolkit[14].

In the last few years, agent technology, especially mobile agents, became a new exciting field in computer science. There exist a huge number of approaches, toolkits, and platforms of different quality and maturity. Therefore, a set of criteria, which collected the requirements to the platforms, is necessary to made. Each agent

platform is evaluated according to the following criteria: Standard compatibility, Communication, Agent mobility, Security, Availability, Usability and Development issue[1].the following table show the comparison results of JADE and Aglet.

| ***JADE*** | ***Aglet*** |
|---|---|
| Open source | Open source |
| Good documentation | Very good documentation |
| Acceptance of users | - |
| - | Clarity in structure |
| Very good GUI | Good GUI |
| FIPA | MASIF |
| Very good security features | Build – in security |
| Weak mobility | Weak mobility |
| Various communication protocols | Communication using sockets |

[ Table 2.2: Comparison JADE and Aglet [1

Mobile agents considered a very interesting technology to develop applications for mobile, pervasive, and distributed computing. So, a common problem when one wants to benefit from mobile agent technology to develop distributed applications is the decision about which platform to use[15]. *Raquel Trillo et.al*, compare the agent toolkits qualitatively and evaluate their performance in a variety of settings with an extensive set of experiments and highlight the importance of this work and justify the choice of the mobile agent platforms that they have considered for comparison as follow: *Aglets*, *Voyager*, *Grasshopper*, *Tryllian*, *JADE*, *Tracy*, and *SPRINGS*. according to the survey in [11], Aglets, Voyager and Grasshopper were among the four best mobile agent platforms; their ranking was: 1) Grasshopper, 2) Jumping Beans, 3) Aglets and 4) Voyager[2]. In general, the different works where some mobile agent platforms compared are partially outdated today, as new platforms and versions have appeared since the publication of such works. For example, as far as we know no other survey evaluates JADE and Tryllian in the context of mobile agents. As developers of distributed systems based on mobile agents, we think that a work with an updated comparison should be beneficial to the community[2].

Raquel Trillo,et al, summarized their results as follow:

| Feature | Aglets | JADE |
|---|---|---|
| Model | Events | Behaviors |
| Elements | -Contexts<br>-Agents (aglets)<br>-Tahiti | -Containers<br>-Main container<br>-Platforms<br>-Agents<br>-DF, AMS, MTS |
| Proxies | Yes | No |
| Dynamic proxies | No | No |
| Synchronous communication | Yes (deadlock) | No |
| Asynchronous communication | Yes | Yes |
| Messages | Yes | Yes (FIPA) |
| Remote calls | No | No |
| Callbacks after movements | No | No |
| Call/messages by name | No | Yes (AID) |
| Movements by name | No | Yes (via AMS) |
| Available for download | IBM Public license | LGPL |
| GUI Tools | Some | Yes |
| Level of activity | Very low | Very high |
| Security mechanism | Basic | Yes (JAAS, etc.) |
| Some other features | -ATP<br>-Itinerary | -FIPA<br>-Jess, JADEX, etc.<br>-Ontology support |

*[ Table 2.3: Qualitative comparison among mobile agent platforms [15*

There are several Java-based mobile agent systems commercially available. These mobile agent toolkits provide all classes needed in Java for building such systems. The builder supplies the agents with the “brain” the algorithms that will be used to accomplish the given goals. The paper give an overview of four agent systems that use Java: IBM’s Aglets, Object Space’s Voyager, General Magic’s Odyssey, and MEITCA’s Concordia.[16-18].

Only Aglet can be consider, because the other toolkits is out of research scope.

| Agent system | Aglet |
|---|---|
| GUI | Yes |
| Modular design | No |
| Mobility mechanism | Sockets |
| Persistence | None |
| Security | Security manager |
| Direct agent-agent communication | Yes |
| Communication type provided | Synchronous, Asynchronous and broadcast |

*[ Table 2.4: Aglet features [16*

JADE offers a more efficient implementation and more general agent model. JADE written in the Java language and comprises various java packages, giving application programmers both ready-made pieces of functionality and abstract interfaces for custom application dependent on tasks. The development of JADE has not yet terminated. Our intention to add the support for agent mobility and offer some higher level agent architecture[19].

(Luis Moura Silva et al, 1999). They present the results of an experimental study where they compare the performance of eight Java-based Mobile agent systems: Aglets, Concordia, Voyager, Odyssey, Jumping Beans, Grasshopper, Swarm and JAMES[20]. Although the study was not consider JADE but it highlight about performance and the robustness of each platform of aglets. Results show that it is a very robust platform and it has passed all the tests without crashing. The performance is not so good when compared with other platforms. Depending on the test cases. The network traffic is also a weak point of this platform[20]. The list of features and the overall functionality of each platform also play a very important role. To get a complete picture about the best platforms the interested reader should still take a look to the list of features that are supported by each system[20].

There exist a huge number of approaches, toolkits, and platforms of different quality and maturity. This fact led us to the problem of evaluating them to find an "optimal" platform. They defined a set of criteria, which collected the requirements to the platforms. While evaluating platforms we found that nearly each one has its own philosophy regarding development of agents. The standardization efforts undertaken by FIPA and MASIF considered only by a few agent development platforms. Substituting one platform by another requires a redesign of all the code[21].

Choosing the right or most suitable platform for a particular application area, base on their performance is a challenge for both the developers and the users. They carried out a qualitative comparison across three selected, Java based Mobile agent System, Aglet Tracy, and JADE[9]. The result showed that aglets are not scalable and provides weak security measures. JADE has a strong security measure, scalable and its multi agent feature will enrich its usage on the internet. Results also shows that it

will be better to use JADE for file transfer and retrieval, and discovered that data compression has a positive effect, The results showed that compressing small amount of data, sometimes increased the size of the data[9].

Although there are several agent-building toolkits that can help developers to effectively develop MAS and allow them to focus more on the application to specific domains, it is difficult for developers to select the appropriate toolkit. Shakshuki presented a methodology to evaluate multi-agent toolkits. The proposed methodology considered availability, environment, development, characteristic properties and performance as the main evaluation criteria. Using this methodology will make it easy for developers to select the appropriate toolkit based on their interest and needs. According to shakshuki methodology, JADE has get percentage 74% and Aglets get percentage 55% of evaluation[10].

(Another research study done by [22]). The study compare five agent toolkits, the results showed that JADE is more attractive that other because it contain the basic and important feature of platform, open source, support API, fully implemented in java, consistency on different internet hosts and has good security feature, also support mobility for agent and compliance with FIPA. While other toolkits support some of feature[22]. The following table show summarize of results.

| Toolkits Features | Aglet | Voyager | JADE | Anvhor | Zeus |
|---|---|---|---|---|---|
| Nature of porduct | Available, open source | commercial | Available, open source | Available on BSD lisence | Available, open source |
| Communication mechanism | Synchronous and Asynchronous | All methods | Asynchronous | Asynchronous | Asynchronous |
| Security mechanism | Weak | Weak | good | Strong security | Good |
| Mobility protocol | Weak | Weak | With some Weak | Weak | Not support |
| Migration mechanism | socket | RMI | RMI | Socket | Not support |
| usability | Electronic markets for plane tickets in Japan | Build of distributed java applications that utilize of different graphical I/O like mobile devices iphone | A number of universities, companies, research centers and laboratories ... | Commercial and medical applications | Companies of commercial and inttelgent systems |

*[Table 2. : Comparison Between Toolkits Used to develop Mobile Agent [22*

## 2.6**Discussion**

In section 2.5, we found there are many of studies submitted in the area of toolkits use for development of mobile agent applications, some of these studies was a survey on the existing toolkits that summarize the language, license and production information while some another studies was compare qualitatively the features, mechanisms, standard follows and nature of the product of each toolkit.

Except the methodology proposed by Elhadi Shakshuki we does found any methodologies applied in these studies even of this methodology applied by two of these studies the first study it was by the author (Elhadi Shkshuki) and the other by (Josef Altmann et. al) this study was not apply the methodology as it specified only consider some of the criteria and it's aspects and the final evaluation does not summarized as a percentage value!

The following table summarize the studies and short comment.

| Author | Title | comment |
|---|---|---|
| نجلاء بديع إبراهيم<br>حليمة عيسى سليمان 2013 | دراسة وتطوير نموذج أمنى للشبكة باستخدام تقنية الوكيل | - Survey<br>-JADE more attractive |
| Oyewole et al 2012 | A Java Simulation-Based Performance Evaluation of Mobile Agent Platforms | -Qualitative comparison based on transmission time and data compressed |
| Aarti Singh et al 2011 | Agent Development Toolkits | - Survey<br>-JADE seems most appealing<br>-Aglet does not comply with FIPA and it has lack of security and scalability |
| Raquel Trillo et al 2007 | Comparison and Performance Evaluation of Mobile Agent Platforms | -Qualitative comparison and evaluation performance in variety of settings |
| E. Shakshuki 2005 | A methodology for evaluating agent toolkits | - Frame based evaluation |

| | | |
|---|---|---|
| | | based on set of criteria |
| Nguyen G. et al 2002 | Agent Platform Evaluation and Comparison," | -Survey<br>-Show JADE in good ranking than Aglet |
| Fabio Bellifemine et al 2001 | Developing multi-agent systems with a FIPA-compliant agent framework | -Introduction to JADE |
| Josef Altmann et al 2001 | Using Mobile Agents in Real World: A Survey and Evaluation of Agent Platforms | -Evaluation based on selected criteria |
| Luís Moura Silva et al 1999 | The Performance of Mobile Agent Platforms | -Benchmark study<br>-Does not consider JADE |

Table 2.: summarization for related studies

## 2.7**Chapter summary**

the chapter classified into six paragraphs: paragraphs one, two and three contain the introduction about JADE, Aglet and the foundation of Agent standard also contain the Agent communication languages. Paragraph four explain the evaluation based on framework which is proposed by Elhadi Shakshuki. Paragraph five about the related studies while paragraph six is discussion about these studies.

# CHAPTER THREE

## Methodology

# Chapter (3)

# Methodology

## 3.1 Introduction

This research, will apply three phases to compare between multi agent systems toolkits. the first one is a **Framework-based** evaluation – will apply the methodology for evaluating agent toolkits which is proposed by [10], the second method is **a code-based** evaluation method and the third method is **other-features** .based evaluation method which is features that distinguish the toolkits each other

.The overall methodology is shown in the figure 3.1 below

Figure 3.: Methodology Of evaltion JADE and Aglet

.Details in each method of the methodology is illustrated in the next figures

## 3.2 Frame-based evaluation based on Shakshuki methodology

As we explained in chapter two section, 2.3 there are five criteria should consider. The next section – criteria matrix - show the evaluation of each of the five .criteria will done

### 3.2.1 Criteria matrix of evaluation

#### 3.2.1.1 Availability

This criterion is based on the following measurement method: analysis of user license and/or sales contract to check the costs and license issues, analysis of toolkit documentation, check the Internet page and determine whether there is technical support. This criterion has weight (10%). and main aspects that are considered in this criterion are:

- *Evaluation version:* developers can use the toolkit for a long or short period of time : for free. this criterion is valuated as
  .NO-1
  .YES, but you cannot utilize the full functionalities-2
  .YES, but you have to register for a key and it is valid for a few days-3

.YES-4

- :*Costs:* developers consider cost as an important factor and valuated as

1-Costs are more than 1000 dollars.

2-Costs are less than 1000 dollars.

3-Limited license.

4-Free for research and education purposes.

- *Development Phase:* is concerned with the development state of the product (alpha, :release, phase-out, application examples) this criterion is valuated as

.(Phase-out or pending (no longer supported or improved-1

.Early beta release-2

.Beta release-3

.Production release-4

- *Technical Support:* concerned with whether support is provided with the toolkit e.g. :news-group and internet support, this criterion is valuated as

.NO support-1

Only FAQ, internet support-2

.News-group-3

.Support available through email requests-4

**Environment**3.2.1.2

This criterion is based on the following: analysis of existing documentation and internet pages where platforms are described, installation of the toolkit to check time and effort required for installation and maintenance by prototype implementation. :This criterion has weight (10%). And the main aspects considered are

- :*Documentation:* this criterion is valuated as

.NO documentation-1

.User guide only-2

.Good programmer's guide-3

.Very good programmer's guide-4

- *Operating System Support:* concerned with the agent environment platform :dependency, and valuated as

.Dependent on operating system-1

.Independent of operating system-4

- *Installation Effort:* concerned with how much effort is necessary to install the agent platform, this criterion is valuated as:
  1-Failed to install.
  2-Difficult to install.
  3-Easy to install.
  4-Very easy to install.

**Development** 3.2.1.3

This criterion is based on the following measurement: study of platform and programming documentation to know the architecture and the implemented language, as well as prototypical implementation to check the issues related to software engineering. This criterion has weight (20%). and valuated as:

- *Programming language:* This aspect is concerned with the programming languages the toolkit supports for the development and the execution. This criterion is valuated as:
  1-Does not support Java.
  4-supports Java.
- *Graphical administration:* This aspect is concerned with whether the toolkit provides graphical administration interfaces for users to create and manage agents. This criterion is valuated as:
  1-NO GUI provided.
  2-Simple GUI, which can view agent's structure.
  3-Good GUI, which can define rules and specify protocols.
  4-Exellent GUI, which can view agent activities.
- *Rapid application development (RAD):* This aspect is concerned with whether the toolkit includes a full featured editor with wizards and help-functions and if it is possible to edit generated code. This criterion is valuated as:
  1-NO support of RAD.
  2-Poor support of RAD with limited function.
  3-Good support of RAD, which helps developers effectively create agent.
  4-Exellent support of RAD, which can easily debug application.
- *Architecture:* This feature defines the design principles such as class structure and services and is concerned with whether the toolkit follows known design principles to develop agent systems. This criterion is valuated as:

.NO suggestion of support for this feature-0

.Does not support this feature-1

Poor support of this feature, which only partially complies with the known-2 .design principle

.Good support of this feature, which complies with the known design principle-4

### 3.2.1.4 **Characteristic properties**

Characteristic properties include communication standards, mobility, coordination and security. This criterion is based on the following measurement: analysis of existing documentation and Internet pages where platforms are described to study coordination, communication and mobility mechanisms. This criterion has weight (30%). And the main aspects considered are:

- *Communication Standard:* A communication standard like Knowledge Query and Manipulation Language (KQML) or Foundation for Intelligent Physical Agents (FIPA) can facilitate the acceptance of MAS toolkits. This criterion is valuated as:

  .No suggestion to support communication standard-0

  .Does not support communication standard-1

  .Poor support for this feature, which partially follows the standard-2

  .Good support for this feature, which fully follows the standard-4

- *Mobility:* Mobile agents can transport themselves from one machine to another, which can reduce network traffic and improve the network performance. This criterion is valuated as:

  .NO suggestion to mobility-0

  .Does not support mobility-1

  .Partially supports mobility-2

  .Fully supports mobility-4

- *Coordination:* Agents with good coordination support can resolve problems quickly, make timely decisions and facilitate achieving the goal. This aspect is concerned with whether the platform provides mechanisms for multi-agent coordination. This criterion is valuated as:

  .NO suggestion of support for this feature-0

  .Does support this feature-1

  .Supports this feature-4

- *Security:* Good security support can make data exchange among agents more reliable. This criterion is valuated as:

.NO suggestion for this feature-0

.Does not support this feature-1

.Supports this feature-4

**Performance**3.2.1.5

This criterion describes the MAS toolkits on the message transport system (MTS) performance, including Intra-process and Inter-process message delivery. The performance of message delivery affects the MAS communication efficiency. Evaluation of MAS toolkits on this criterion is based on prototype implementation on Intra-process and Inter-process agent communication to check whether the toolkit supports agent communication and to test its performance and scalability. This criterion has weight (30%). the main aspects considered are as follows:

- *Intra-process Message Delivery:* Intra-process refers to the process in which all agents in MAS are living in the same process. All the implemented toolkits support Java, and this criterion is valuated as:

.Failed to implement prototype-0

.Bad support of this feature-1

.Poor support of this feature-2

.Good support of this feature-3

.Excellent support of this feature-4

- *Inter-process Message Delivery:* Inter-process refers to the process in which all agents in MAS are living in different processes. This criterion is valuated as:

.Failed to implement prototype-0

.Bad support of this feature-1

.Poor support of this feature-2

.Good support of this feature-3

.Excellent support of this feature-4

## Scenario-based evaluation3.3

In this phase, we will implement same scenarios on both JADE and Aglet. The objective of these scenarios is to evaluate the following (communication between agents, how security implemented in each toolkit, migration and cloning of agent).

:More details on these scenarios is shown as follow

### 3.3.1 **Communication between agents**

We will build an agent-based system consist of two agent **sender agent** that send message –to get information about some file in another host - to **receiver agent** .that reply with another message -weather the file exists or not

### 3.3.2 **Security**

Both of JADE and Aglet follow a specific method to apply security either through *policy file* or through *additional package*. Therefore, we will evaluate the security in each toolkit based on the implementation. Although there was security .consideration not implemented on both toolkit

#### 3.3.2.1 **Encryption of message**

We will build two an agent based system sender agent that send an encrypted message to receiver agent. To evaluate how the toolkit dealing with the encryption of .message

#### 3.3.2.2 **Authentication and permissions**

We will create new user and give it the permission to execute some task in .another host. To evaluate how the toolkit deal with other incoming services

### 3.3.3 **Mobility features**

#### 3.3.3.1 **Migration of agent**

We will build an agent-based system that migrate to another host to extract some information from a file there. To evaluate which toolkit have supported classes .of creation, transferring, retracting and dispose (kill) of the mobile agent

#### 3.3.3.2 **Clone of agent**

In this case, we will build an agent-based system that clone itself in some host and return information when needed. To evaluate which toolkit have supported .classes to make the agent clone itself

## 3.4 Other-features based evaluation

For each toolkit there are some features that distinguish it , but these features cannot evaluate using Shakshuki–Methodology and cannot be illustrated by doing code scenario, so that will mention these features according to the using of both toolkits and the evaluation will be according to the researcher point of view based on the use.

## 3.5 Chapter summary

This chapter explained, how the comparison will done. So its contain the information about the three phases. phase one explain how the results will obtained while apply the framework proposed by Elhadi Shakshki, phase two explain the method of evaluation while running the same scenarios while phase three explain the method will follow to obtain the results based on the other-features method of evaluation.

# CHAPTER FOUR

## Results and Discussion

# Chapter (4)

# Results and Discussion

## 4.1 Introduction

The results obtained in this chapter is based on different phases as is explained in chapter three section 3.2 , 3.3 and 3.4. The result obtained from phase one is calculated and summarized as percentage value, which calculated from five criteria -according to Shkashuki methodology-. The results obtained from phase two will illustrate the benefits or advantages of each tool in terms of ease of use and ease of application of some of the functions used in the development of agents. While the results obtained in the phase three, will summarize the features found in each tool that facilitate of using toolkit and designing mobile agent.

## 4.2 Frame-based evaluation results

This section describe the evaluation of JADE, and AGLETS based on Shakahuki methodology. Five criteria specified for this evaluation is availability, environment, development, characteristic properties, and performance. The following paragraphs show the comparison results based on these criteria.

### 4.2.1 Availability

The following table illustrate the availability aspect the weight assign for this aspect is 10% from the total percentage, for optimal evaluation, each toolkit must be obtained 16 marks in the total. The following describe the mark obtained:

**Evaluation version**: Both JADE and Agent has get 4 marks because they can be use and download from the website freely.

**Cost:** Both JADE and Agent has get 4 marks because they are an open source and can be use freely.

**Development:** JADE has get 4 marks because it has continuously update versions while Aglet has get 2 marks because the last version was before 2005.

**Technical support:** Both has get 2 marks because they has a support via FAQs on the internet.

| availability 10% | | | | | |
|---|---|---|---|---|---|
| | Evaluation version | costs | development | support | total |
| **Aglets** | 4 | 4 | 1 | 3 | 12 |
| **JADE** | 4 | 4 | 4 | 4 | 16 |

*Table 4.1: Availability aspects*

*Figure 4.: Availability aspects*

### 4.2.2 **Environment**

The following table illustrate the Environment aspects the weight assign for this aspect is 10% from the total percentage, for optimal evaluation, each toolkit must be obtained 14 marks in the total. The following describe the marks obtained

**Documentation:** JADE has get 4 marks because it has a good documentation that contain the documentation for programmer and documentation for administration .while Aglet has get 3 marks because it's documentation is programmer guide

**Support O.S:** Both has get 4 marks because they are an independent operating .system toolkits

**Installation Efforts:** JADE does not need other software for installation so it's get 4 marks while Aglet needs for additional software called *Appachi Ant* for install so it's .get 3 marks

| environment 10% | | | | |
|---|---|---|---|---|
| | documentation | supported OS | installation effort | total |
| **Aglets** | 3 | 4 | 3 | 10 |
| **JADE** | 4 | 4 | 4 | 12 |

*Table 4.2: Environment aspects*

*Figure 4.: Environment aspects*

### 4.2.3 **Development**

The following table illustrate the Development aspects the weight assign for this aspect is 20% from the total percentage, for optimal evaluation, each toolkit must be obtained 16 marks in the total. The following describe the marks obtained

**Programing Language:** Both JADE and Aglet has get 4 marks because they support .Java language

**Graphical Administration:** JADE has get 4 marks because it has more additional packages (*Add-ons*) used for monitoring the agent while Aglet has a simple GUI for .monitoring the agent

.**RAD:** Both of them has get One mark because they does support RAD

**Architecture:** JADE has get 2 marks because it's follow a methodology for .developing an agent based systems, while does found any method follow by Aglet

| **development 20%** | | | | | |
|---|---|---|---|---|---|
| | Programming language | graphical Administration | RAD (Rapid Application development) | Architecture | total |
| **Aglets** | 4 | 3 | 1 | 1 | 9 |
| **JADE** | 4 | 4 | 1 | 2 | 11 |

*Table 4.3: Development aspects*

*Figure 4.: Development aspects*

### 4.2.4 **Characteristic properties**

The following table illustrate the Characteristic properties aspects the weight assign for this aspect is 30% from the total percentage, for optimal evaluation, each toolkit must be obtained 16 marks in the total. The following describe the marks :obtained

**Communication Standard:** JADE has get 4 marks because it's compliance with .FIPA specification, while Aglet was follow the MASIF communication method

**Mobility:** JADE has get 2 marks because it support mobility partially, while Aglet .has fully supported for mobility

**Coordination:** JADE has get 4 marks because it support coordination via Contract .net protocol, while Aglet does not follow specific method for agent coordination

.**Security:** Both of them has get 4 marks because they support the security

| **characteristic 30%** | | | | | |
|---|---|---|---|---|---|
| | communication Standard | mobility | coordination | security | total |

| | | | | | |
|---|---|---|---|---|---|
| **Aglets** | 2 | 4 | 1 | 4 | 11 |
| **JADE** | 4 | 2 | 4 | 4 | 14 |

*Table 4.4: Characteristic properties aspects*

*Figure 4.: Characteristic properties aspects*

**Performance**4.2.5

The following table illustrate the performance aspects the weight assign for this aspect is 30% from the total percentage, for optimal evaluation, each toolkit must be obtained 8 marks in the total. The following describe the marks obtained

**Inter-process and Intra-process:** Aglet does not support these features, all agent in aglet they live in the same place, while JADE has support intra-process and has an agent called AMS(Agent Mobility Services) that implement the intra-process, also JADE has support the inter-process through additional package.

| **performance30%** | | | |
|---|---|---|---|
| | inter-process | intra-process | total |
| **Aglets** | 0 | 0 | 0 |
| **JADE** | 4 | 4 | 8 |

*Table 4.5: Performance aspects*

*Figure 4.: performance aspects*

In order to calculate the percentage of each toolkit, some equations used for this purpose as they described in Shakshuki methodology.

**Calculation of the  for each criteria catalogue.**

 = the sum of the points that the criteria catalogue can obtain it multiplied by the weight that was assigned for the criteria catalogue. The formula to calculate the *ci* is:

From the above results, we can calculate the *Ft*, which is the feature sum of toolkit *t.* by the following formula:

= 7.3

= 11.6

The *Fa* for each toolkit is 13.2.

The percentage for each toolkit calculated using the following formula:

Then:

*Aglets (Pt) =  = 55%*

*Whereas, JADE (Pt) = = 88%*

The following table describe the previous result in summarized form:

| toolkit | Fa | Ft | Pt |
|---|---|---|---|
| Aglets | 13.2 | 7.3 | 55% |
| JADE | 13.2 | 11.6 | 88% |

Table 4.6: Final remark of JADE and Aglet when apply shakshuki Methdology

## 4.3 **Discussion**

There was imparity in the results when the methodology is implemented by Shakshuki in 2005 – Aglet get 55% while JADE was get 74% already shown in chapter two – we think this variation case by the update of JADE, especially JADE does not support security issues and there is no architecture specified. The results show that aglet does not show any progress because there is no update till now.

## 4.4 **Scenario based evaluation results**

### 4.4.1 **Agent communication**

That results show that the communication between agents in JADE is more efficient than AGLET. Because JADE is compliance with FIPA specifications so that allow it use FIPA ACL – which provide separation between peformative and content of message – this mechanism allows agent send message using different languages, the receiver agent analyzing the message performative and reply depend on this performative (it is already specified how to reply).

The following segment of code illustrate the buyer agent that send a Call For Proposal message (CFP) to many of agent (i) and wait for their proposal.

*Figure 4.: part of code illustrate using CPF message*

Here the buyer agent receive the PROPOSE message that sent by seller agent

*Figure 4.: part of code illustrate receiving PROPOSE message*

Another thing JADE is use DF concept: is the agent that implements the yellow pages service. used by any agent wishing to register its services or search for other available services - this concept allow agents to declare their services – other agents can search in DF by the type of service that it need and communicate with agent/s .who provide this service

The following segment of code illustrate the **seller** agent that declare of his name and .type of service

*Figure 4.: part of code illustrate the registration the agent name and the service in the DF*

The following code illustrate the **buyer** agent using DF to agents who provide **.selling**

*Figure 4.: part of code illustrate agent search for seller agent in the DF*

Whereas AGLET is used the MASIF standardization, so that any message has one parameter Message kind which specify the type of message that the receiver will .extract it and reply base on

The following segment of code illustrate a Seller agent that receive messages from .the buyers

*Figure 4.: part code Aglet code illustrate how to seller agent will reply*

The method *m.sameKind()* represent the communication method, in this method the .**seller** agent analyze if **buyer** ask with specified questions and sent reply

In this method, the agent must exactly know the name and location of the agent that needs to communicate with and there is no method for agent show their service that .provided also there is no method to extract the agent name from the message object

### **Security**4.4.2

The security was implemented in JADE through additional package must installed with basic classes of JADE in order to use it; the package include the authentication, authorization, agent permission and message signature and encryption. So that all platforms need to install this package. While the security in .AGLET is built in

#### **Authentication**4.4.2.1

The authentication in JADE provides a guarantee that a user starting a JADE platform and thereby generates containers and agents within that platform. It consist of two elements:

**CallbackHandler,** which allows the user to provide its username and password.

**LoginModule,** which checks validity of username and password.

The following command illustrate the starting the secure main container from the command line:

```
java jade.Boot –conf main.conf –gui
```

The **–** ***conf*** command tell the JADE runtime there are some setting in **main.conf,** which is a file**,** contain the settings.

Figure 4.: setting of the security of strating secure main container based on JADE

The following command illustrate the attach new container with main container

```
java jade.Boot –conf cont-1.conf –container… -host… -port 1099
```

The command ***-conf cont-1.conf*** contain some settings for the attached container:

Figure 4.: Settings of attach new container with main container

The following GUI will appear after writing the above command for authentication pocess.

Figure 4.: attach container confirmation based on JADE

Before you attach new container the platform you want to pass the authentication , the username and password must be already give the permission that allow to pass this check. In the security folder will find the text file (password.txt)

Figure 4.: Username and Password setting based on JADE

The file contain the user name and password. But it's not enough to pass the check the username must give some permissions (see permission in the next).

The authentication in AGLET for starting the Tahiti GUI only so that any agent can access the Tahiti but the control will be over the permissions.

Then when you try to start Tahiti, a window will appear to insert the username and password

Figure 4.: Starting Tahiti GUI based on Aglet

The user must pass this check to start up the Tahiti, the username and password must specified in the first steps of aglet installation process.

**Permission**4.4.2.2

Both JADE and AGLET implements the permission through some settings in *policy file* and the security method will check this file to specify if the agents has .permission to execute the task or not

.Here is some permissions that gave to user JHD

Figure 4.: Permission Setting Based on JADE

While the policy file AGLET was located in the HOME windows folder

For example: C:\Users\siu\.aglets\security\aglets.policy contain permissions for ***siu*** user

Figure 4.: Permission Setting Based on Aglet

Therefore, if the aglet does not have permissions to access specific location or services can show the following error.

Figure 4.: Error Security message

**Message encryption**4.4.2.3

Because the agents can send the message through network this message will facing some security issue so that JADE has provide some services to encrypt and singing the message it has a methods setUseEncryption() and setUseSignature().

The segment of code illustrate the using of signature in JADE, it done by call the setUseSignature() method and passing the message which you want to sign it as parameter.

*Figure 4.: part of code illustrate signing the agent message*

The segment code illustrate the using of encryption in JADE, it done by calling the setUseEncryption() method and get the receiver identification in order to get the receiver public key that use to encrypt a message.

*Figure 4.: part of code illustrate the enryption of the agent message*

Although the encryption slow down the performance of agent communication! but sometimes it is important.

Whereas we did not found that AGLET implements security issue that guarantee for sending secure message in **Aglet v2.0.2**.

### Mobility features 4.4.3

#### Migration of agent 4.4.3.1

The mobility of agent is key feature of mobile agent that distinguish it from traditional program.

JADE has provide three ways to dispatch the agent to new host.

By using CMD line (a.

The way require knowing of the address (container) which agent will travel to.

```
java jade.Boot –container –container-name name -host IP -port 1099 …..
```

Write code (b.

This way is use when the agent decide to travel autonomously.

The method *domove*( ) make the agent to travel to the destination it require specify the name of container (host) that agent will travel to as a parameter.

*Figure 4.: part of code illusrate of moving the agent to another location*

Use can specify the destination static or dynamic from information of agent that is communicate with.

The segment of code illustrate the extracting of the sender information

*Figure 4.: part of code illustrate of extracting the information of sender of message*

The Figure below show the results of executing the above code

Figure 4.: Receiver agent extract information of sender agent

Use GUI (c.

Also JADE has provide simple interface allow user to dispatch the agent to another location. The way require user to know what is name of container exactly.

Figure 4.: Dispatching Agent Using GUI Based on JADE

Also Aglet has a good mobility features. It used two way to dispatch agent to another .location

Write code(a

In order to dispatch an agent to new location you just need to call *dispatch()* method .and specify the parameter which is a URL of destination

.The following code illustrate the dispatching of agent to new host using Aglet

*Figure 4.: part of Aglet code illustrate of moving the agent to another location*

Use Tahiti GUI(b

The Tahiti has GUI the allow user to dispatch agent to another destination in addition .has service that allow to return back the agent to the home

**Dispatching of agent**

Figure 4.: Dispatching Agent Using GUI Based on Aglet

**Return an agent from another host.**

Figure 4.: Return back Agent Using GUI Based on Aglet

However, here is a common problem is that the user can return any agent from that !host

**Clone of agent**4.4.3.2

Both JADE and AGLET has services that allow user to clone of agent. They provide good classes and the process of clone agent is very simple. Because the clone it is like dispatching with additional parameters the new name of agent and the .location

## 4.5 Discussion

The scenarios, implementation and setting in the above sections are summarized in the following table. The table illustrate each feature and the comparison issue between JADE and Aglet.

| Feature | | JADE | AGLET |
|---|---|---|---|
| Communication | | **Very Good :**<br>Follow standard for agent communication it compatible with FIPA standard | **Weak:**<br>Does not follow standard they use paradigm for agent interoperability (MASIF). Because The aglet created when the FIPA is proposal. But not updated till now |
| Security: | Authentication- | **Very good** mechanism. Before attaching any container must be authenticated | **Weak mechanism.** Authentication for starting the Tahiti interface. So any agent can access the Tahiti |
| | permission- | **Good**: Each agent must to be authorized | **Good**: Each agent must to be authorized |
| | message encryption- | Have services to encrypt and sign of message | Does not support |
| Mobility | Dispatch of agent | Very good and simple method to dispatch the agent | Very good and simple method to dispatch the agent |
| | Optional method for dispatch agent | **Yes**:<br>It has an optional services execute when the start to move | **Yes**, very good services |
| | Return back agent | **No GUI** services **Code only** | **Code and GUI** |
| | Clone of agent | **Very good** services and simple method for clone agent | **Simple** method for clone agent |

| | | | |
|---|---|---|---|
| | Optional method for clone | **Yes:** there are some optional method execute when agent start to clone it self | **Yes very good** with more details |

Table 4.7: criteria of mobile agent that evaulated in this research

## 4.6 Other-Features based evaluation results

Some different optional features have founded when using both JADE and Aglet, some of these features have different usage are depend on the application. The following table summarized these features and compare between these two toolkits depends on some feature.

| Feature | JADE | AGLET |
|---|---|---|
| Additional packages | Have ability to integrate with more additional packages | Does not find any additional package |
| Support lightweight devices | Support running on devices running Android and Microsoft.Net Framework | Does not support |
| Monitoring of agent | Have services that show when agent what doing when it start his task. Like (sniffer (agent | Does not found services that monitor the agent |
| Agent naming | Agent name can be freely for user. | Class name is name of agent |
| Connect platform together | In order to communicate all JADE platform must be connected together within one main container | Network infrastructure only, because it use IP in communication |
| Scalability | Sometimes does not work properly after installation of some additional packages! | Does not effects |

Table 4.8: Features of JADE and Aglet

## 4.7 Discussion

The results show that the GUI of JADE is better than the GUI of aglet because JADE has flexibility in installation of additional package and ability for installation on limited resources environments. also JADE has GUI that include components for management the agent and the messages also for monitoring the agent, while aglet GUI include the basic components for agent management.

## 4.8 Chapter summary

This chapter describe the analysis based on the methods of evaluation, the chapter include analysis and discussion for each method.

# CHAPTER FIVE

## Conclusion and Future work

# Chapter (5)

# Conclusion and Future work

## 5.1 Conclusion

A comparative study between JADE and Aglet has been submitted, the comparison was based on three phases of evaluation.

In phase one we used a framework of evaluating the Agent toolkits, the results obtained in this phase represented as a percentage values the results obtained showed the preference for JADE than Aglet. JADE has get 88% while Aglet has get 55%.

Phase two: the evaluation based on running same scenarios on this toolkits,three criteria has been selected in evaluation the criteria was (communication, security and mobility), the results showed that the communication in JADE is better the the communication in Aglet. The security criteria in JADE is better than Aglet because its contain many of security features as we metioned in section (). In the mobility criteria the showed the preference of Aglet versus JADE.

While the results obtained in phase three showed JADE has many features for management and monitoring the agent inclded in its GUI while Aglet has simple GUI.

## 5.2 Future work

- Apply Shakshuki Methodology to compare other toolkits of agent-based application.
- Compare JADE and Aglet using another Methodology for evaluating agent based application with different criteria.
- Propose a methodology to evaluating agent based application using more than one method of evaluation.

## 5.3 **Chapter summary**

Better mobile agent with a good communications paradigm with a good Security services is achieved using JADE because it compatible with FIPA standards, it has security mechanism for sending message and permissions of agent, good support available with community of jade, up to date continuously and more additional add-ons are available.

Aglet it not bad toolkit and it has good mobility services which is enough criteria for developing mobile agent application, but when using Aglet may result in missing some important features like security and communications and standardization.

## REFERENCES


[1] D. T. T. Nguyen G., Hluchy L., Laclavik M., Balogh Z.and Budinska I.,, "Agent Platform Evaluation and Comparison," *Published by Institute of informatics, Slovak Academy of Sciences*, June 2002.

[2] S. I. a. E. M. Raquel Trillo, "Comparison and Performance Evaluation of Mobile Agent Platforms."

[3] J. Wooldridge, "Intelligent agents: theory and practice," *The Knowledge Engineering Review*, 1995.

[4] Ovum, "Intelligent agents: the new revolution in software," 1994.

[5] P. M. Sri Raj Dutt, Sourav Khandelwal, "Mobile Agent," *Summer Project Report*.

[6] M. O. Danny B. Lange, "Programming and Deploying Java Mobile Agents with Aglets," 1998.

[7] G. S. L.M. Silva, P. Martins, V. Batista, L. Santos, "Comparing the performance of mobile agent systems: a study of benchmarking," *Computer Communications 23*, 2000.

[8] G. K. Ritu Gupta, "A Survey on Comparative Study of Mobile Agent Platforms," *International Journal of Engineering Science and Technology (IJEST)*, vol. 3, 2011.

[9] A. S. Oyewole, Osunade, O & Azeez, N.A., "A Java Simulation-Based Performance Evaluation of Mobile Agent Platforms," *Computing, Information Systems & Development Informatics*, 2012.

[10] E. Shakshuki, "A methodology for evaluating agent toolkits," in *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, 2005, pp. 391-396 Vol. 1.

[11] A. J. K. Ludwig, W. E. Edgar, S. Wolfgang, and G. Franz,, "Using mobile agents in real world: A survey and evaluation of agent platform," May 2001.

[12] G. C. Fabio Bellifemine, Dominic Greenwood, *Developing Multi-Agent Systems with JADE*, 2007.

[13] *Aglet Home Page*. Available: http://aglets.sourceforge.net/

[14] D. J. Aarti Singh, A.K. Sharma, "Agent Development Toolkits," *International Journal of Advancements in Technology*, 2011.

[15] S. I. a. E. M. Raquel Trillo, "Comparison and Performance Evaluation of Mobile Agent Platforms," 2007.

[16] D. C. Damir Horvat, Veljko Milutinovic, Petar Kocovic and Vlada kovacevic, "Mobile Agents and Java Mobile Agents Toolkits," 2000.

[17] A. W. P. IBM Aglets Workbench, IBM Tokyo Research Laboratory, Tokyo, Japan, 1996.

[18] K. Shah, Guota, R., Timm, S.,, "Study of Mobile Agent Systems," *Department of Computer Science, Virginia Tech, Blachsburg, Virginia, USA*, 1998.

[19] A. P. a. G. R. Fabio Bellifemine, "Developing multi-agent systems with a FIPA-compliant agent framework," *Published in Software-Practice and Experience*, 2001.

[20] G. S. Luís Moura Silva, Paulo Martins, Victor Batista, Luís Santos, "The Performance of Mobile Agent Platforms," *IEEE computer society*, October 1999.

[21] F. G. Josef Altmann, Ludwig Klug, Wolfgang Stockner, Edgar Weippl, "Using Mobile Agents in Real World: A Survey and Evaluation of Agent Platforms," *ACM Press*, May 2001.

[22] ح. ع. س. نجلاء بديع ابراهيم, "دراسة وتطوير نموذج امني للشبكة باستخدام تقنية الوكيل," مجلة الرافدين لعلوم الحاسوب والرياضيات 2013.

[23] O. a. O. S. A. Osunade, "Effect of data compression on Network-Load and Transmission-time in Aglets mobile agent," *Journal of Technical Technology and Vocational Educators (JOTTAVE) University of Calabar, Nigeria*, vol. 2 No. 1 pp86 – 93, June 2011.

[24] S. L. Lytinen and S. F. Railsback, "The evolution of agent-based simulation platforms: A review of netlogo 5.0 and relogo," in *Proceedings of the Fourth International Symposium on Agent-Based Modeling and Simulation*, 2012.

[25] M. Laclavík, *et al.*, "Agent-based simulation platform evaluation in the context of human behavior modeling," in *Advanced Agent Technology*, ed: Springer, 2012, pp. 396-410.

[26] M. Higashino, *et al.*, "Mobile agent migration based on code caching," in *Advanced Information Networking and Applications Workshops (WAINA), 2012 26th International Conference on*, 2012, pp. 651-656.