MAGR: INTEGRATING MOBILITY OF AGENTS WITH ORGANIZATIONS

M.Saber and J .Ferber
1Laboratoire d'Informatique, de robotique
et de microéléctronique de Montpellier
161 rue Ada 34392 Montpellier Cedex 5– France
mansour@lirmm.fr , *ferber@lirmm.fr*

ABSTRACT

This paper presents an organizational model which supports mobility of agents. In this model the mobility is treated as a physical move of the agent and also as a social change of appurtenance. It is a correspondence between an agent organizational model and a mobile agents' standard. We use a script language based on the Itinerary Algebra to represent mobile agents' activity.

KEYWORDS

Mobile agents, organizational model, AGR, Madkit.

1. INTRODUCTION

Nowadays we are facing an important increase of services and information offered to millions of users wherever they are and whenever they want, thanks to a great number of devices such as mobile phones or PDA and environment like Internet, Grid... This new situation, combined with the evolution of the distributed application concept and the maturity of Multi-Agent System paradigm, made researchers think that mobile intelligent agents would be one of the best solution to deal with resulting problems such as dynamic aspect of data and services, reducing the network latency...The problem is that most of researches, dealing with the mobile agents: Aglets (Aglet), TCL (Robert, S. G., 1998), Telescript (White.J., 1996), CLAIM (El Fallah.S.A and al 2003), focused more on the feasibility of mobility for intelligent agents than on providing a real conceptual model. There is no real model dealing, at the same time, with the social aspect of the agents' life cycle while they are in their birth environment or in a foreign one. We believe that providing mobility facilities for agents must be associated to a conceptual model of organization issued from what it was already done by the MAS community. If we take a look at what has been done in terms of organizational models for MAS, we easily remark that all existing approaches such as (AGR, Opera, MOISE…), do not support mobility aspect. The existing models or methodology AUML (Poggi. A., &al, 2004), GAIA, do not provide any organizational solution for designing and administrating mobile agents in an agent society. In this work we address these weaknesses and we prove that it is possible to integrate aspects, organizational social MAS and mobility of agents, into the same model. We extended the concepts of AGR (Ferber and Gutknecht 1998) model to conform those proposed by MASIF standard specifications (MASIF). We implemented the proposed model in the MADKIT platform (Guteknecht and Ferber 2000), in order to offer mobility facilities for users. We propose a script language, based on itinerary algebra (Loke, S.W. and Zaslavsky, A., 2003), for users so they could easily design mobile agents' programs. This language facilitates the creation of agents' itineraries using the primitives we added in the MADKIT platform. The remainder of the paper is organized as follows: Section 2 introduces the model proposed by the Consortium MASIF for mobile agents. Section 3 presents the AGR model and discusses why this model can't support mobile agents' aspect. Section 4 presents the proposed model, which is a combination of the AGR model, and the pyramidal model proposed by MASIF. In section 5 we describe the script language and we present an example for illustration in section 6. Section 7 concludes with a summary of the results and further research.

2. MASIF CONCEPTS

MASIF is a standard for mobile agent systems which has been adopted as an OMG technology. It is an attempt to standardize the mobile agents' technology that tries to facilitate interoperability between agent systems. MASIF proposes to standardize some areas of mobile agent technologies such as management, transfer, naming of agents and agent systems, and location syntax. The definition of what is a mobile agent proposed by the Consortium is as follow: a mobile agent is not limited by the boundaries of the system where he was born (started its execution). It has the capability to move from one network system to another. The state of the agent (execution and attributes) and its source code are also moved with it during this operation of mobility (MASIF). The standard also proposes an organizational model for mobile agents systems.

Region: A region is the association of agencies that are working under the same authority (that could be system or human). A host can receive one or many agencies that could be of the same type or heterogeneous.

Agency: The agency is simply the MAS where agents are running. The agencies have the same definition of the classic platforms for agents plus the capabilities allowing agents to move. These new services consist on deleting agents from local list of agents, serializing it and physically moving it to its destination agency.

Place: The place is the context where an agent can be executed. This environment is determined by an address and can accept one or more agents.

This hierarchical structure facilitates, by its simplicity, the comprehension of the general organization of the mobile agents systems. The problem is that it doesn't specify how communications are organized between agents? How new concepts such "place" must be introduced in classic MAS? How to locate agents?

3. THE AGR ORGANIZATIONAL MODEL

An organizational model is an abstract level where we describe what structures and what interactions will take place in MAS. The implementation of the organizational level by agents is known as the agent level. In this level each type of agent depending on its type (reactive, cognitive), on its behavior, goals and rules will interpret the laws and specifications of the social level. In this section we present the main concepts of AGR model known as Aalaadin (Ferber and Gutknecht 1998). The AGR model is based on three primitive concepts, Agent, Group and Role that are structurally connected and cannot be defined by other primitives.

Agent: an agent is an active communicating entity, playing *roles* within *groups*. An agent may hold multiple roles, and may be member of several groups. One of the most important characteristics of the AGR model is that there are no constraints imposed about the internal architecture of the agent neither on its behavior nor on its capabilities. This is important to make the model as generic as possible so that any MAS could use and adapt it at the agent level. We used this aspect to extend the model so that mobility of agents could be supported and described at higher levels (social level).

Group: a group is a set of agents sharing some common characteristics. A group is used as a context for a pattern of activities, and is used for partitioning organizations. The fact that an agent can belong to several groups at the same time offers more flexibility for applications designers, but as we will explain later this feature could not be accepted for mobile agents. In fact, when we say "belong to a group" we mean logically belonging, but with mobile agents the sense is not the same we will talk about "physically appurtenance" many problems must be resolved.

Role: the role is the abstract representation of a functional position of an agent in a group. An agent must play a role in a group, but an agent may play several roles. Roles are local to groups, and an agent must request a role. Several agents may play a role. So a role is an abstract notion that one or many agents can play. The role as abstraction offers multiple interpretations and implementations at the lower levels (agent level). We used the role notion to express the concept of service that is usually mentioned while speaking about mobility of agents. A mobile agent moves from one place to another in order to request services that are not provided in its place. We will explain later how we introduce this notion in our new model.

Although the AGR model is a generic one, it can not represent some of mobile agents' concepts. For instance the concept of agent doesn't take into account the notion of mobility, neither that of ubiquity. These two aspects are not internal properties and must be considered in the model. When a static agent becomes a member of a group in the AGR model this is a virtual membership, while when a mobile agent moves from one place to another it physically becomes a member of the place.

4. MAGR INTEGRATING MOBILITY TO AGR

4.1 Description of MAGR

In this section we present an extension of the AGR model that takes into account the mobility aspect of agents. This extension, called MAGR, for Mobile-Agent-Group-Role, is based on the idea that agents can physically move from one agency to another. Agents in the model may be of two types static and mobile. Agents may belong to static or a mobile group. Mobile groups will be called places, and static groups represent classic AGR groups. Places propose services that are necessary for a mobile agent to move and perform actions. Agents belong to groups and play roles. Because Roles are the main way through which an agent can act and interact in a space, it is necessary for a mobile agent to have the ability to play a role even if it moves physically from the place that contains the role. This is possible through persistent role.

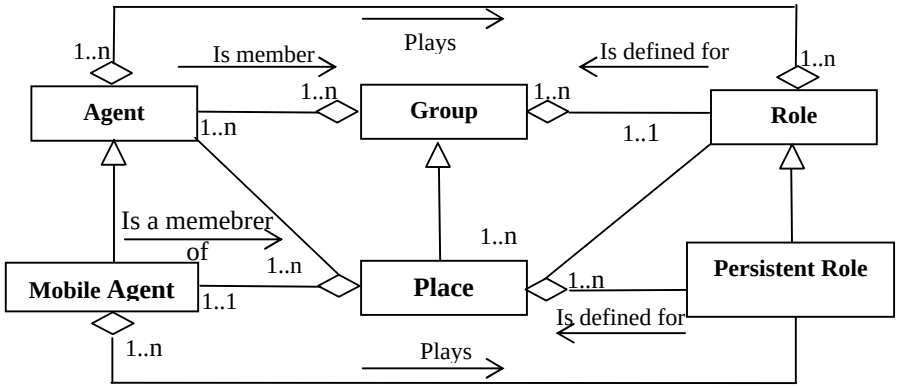

Figure 1. The MAGR model

A classic AGR role could be played only by static agents or mobile agents that are still in the place whereas persistent roles could be played by mobile agents wherever they are. When a mobile agent plays a role, it specifies if it is a persistent role or not. Once the mobile agent moves, all the skills associated to the persistent role will be available. It will be able to send and receive messages to and from agents playing this role. Figure 1 shows the simplified UML diagram, which represents the relations between agencies, groups, places, agents, mobile agents, roles and persistent roles.

4.2 The MAGR Architecture

We adapted the architecture used to implement the AGR model in Madkit platform so that it supports also the MAGR model. The concepts of Agency, Place or Mobile Agent are added to the current platform. An agency contains one or more places. A place hosts both mobile and statistic agents. Let's explain these concepts.

4.2.1 Mobile Agents

The mobile Agent in the MAGR model extends the agent concept presented in AGR with the capability to move. We can define mobile agent as an autonomous entity that communicates with its environment and neighbourhood, plays roles in groups and moves from one place to another in order to achieve its goals.

4.2.2 Agency: Kernel Supporting Mobility

An agency is simply the multi-agent-system in a conventional sense (the platform), extended with services facilitating the migration and social life of the mobile agents. If an application in the platform needs the mobility service it will simply activate it. Agent' mobility presents distinct challenges to the platform: creating, joining places and providing security mechanisms for both agents (mobile and stationary) and hosts (places and agencies). In order to satisfy these requirements a special agent called "AgencyKeeper" is created.

4.2.3 Place: Group able to Host Mobile Agents

Like group, a place is a collection of agents, and the only main difference is that a stationary agent can be a member of many groups at the same time, whereas a mobile agent must join exactly one place at a moment. The relation between a Place and a mobile agent is physical, whereas it is virtual between an agent and a group in the AGR model. So we can say that Place does look like Group but Place should offer a set of services. Place should take into account the whole requirements of the group concept as designed in the AGR model plus the new requirements of mobility. On one hand a mobile agent must be able to have the same capabilities of any stationary agent. In addition, a place must provide discovery services for mobile agents, so they can easily get information about possible destinations. Moreover, the security aspect must be treated by place in order to avoid the intrusion of malicious mobile agents. Finally a Place must publish the services provided by agents (mobile or static) it contains. We create a special agent: the “PlaceKeeper” to achieve these goals

4.2.4 Persistent Roles

A mobile agent moves from one place to another to request services. Places host mobile agents offering services that are requested by other agents. We think that it is natural to consider services as a kind of generic concept of roles. The main problem is that a mobile agent can move at any time and its address changes consequently. To address this new requirement we distinguish two kinds of roles that can be played by mobile agents: the persistent and the non-persistent roles. A Persistent role is designed to make it possible for the mobile agents to receive messages in relation with it. When a mobile agent plays a persistent role, regardless where it is situated it will receive messages in relation with this role. A non-persistent role is simply a classic role. When a mobile agent playing a non-persistent role leaves the place, it will be automatically deleted from the list of the agents playing this role. Managing persistent roles and keep track of the mobile agent position at any time is a complex task. We designed a specific agent whose mission is to address these problems and that offers other services to the mobile agents.

4.2.5 The Mirror Agent

An agent in Madkit has an unchangeable address assigned by the kernel when it is created. Once the mobile agent migrates to a new agency, it will obtain a new address by the host kernel. In order to localize agents continuously, a mobile agent in our solution has two addresses (birth address and current one). The mirror agent is the substitute, of the mobile agent in its birth agency when the latter is in migration. The current address of the mobile agent will be assigned to "NULL" and the mirror agent will be assigned the birth address of the mobile one. The main role of mirror agent is to forward messages, such as those coming form persistent roles, to the mobile agent when it is outside its birth agency. Before moving the mobile agent sends to its mirror agent the list of persistent roles it plays in the foreign agency. When receiving this list, the mirror agent requests to play the same roles as a distant agent. By doing that the mirror agent will receive all the messages related to the persistent role and forward them to the mobile agent wherever it is located. In fact, when the mobile agent migrates from one agency to another it informs its mirror about its next destination. When the migration succeeds the mobile agent sends its new current address to its mirror.

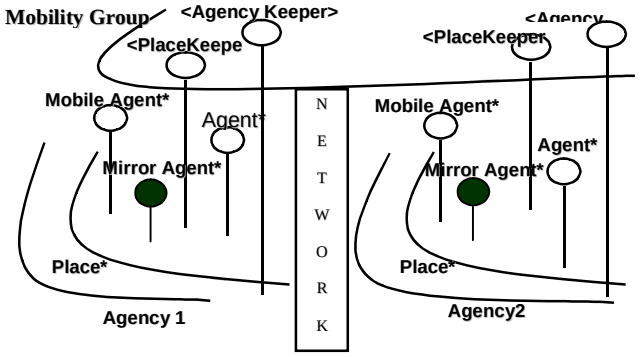

Figure 2. The MAGR architecture

4.3 The Migration Process

Like the majority of the contemporary mobile agent systems we rely on JAVA's object serialization mechanism to transfer agents. The typical MAGR migration process follows these steps:

1. The mobile agent requests the place keeper to authorize it to leave the place.
2. The place keeper verifies if the mobile agent can securely leave the Place.
3. The place keeper asks the agency keeper to verify and validate this migration
4. Agency Keeper accepts or refuses the request of migration and answers the place Keeper.
5. If the place and agency keeper accept the mobile agent departure, the former will ask the foreign place keeper (the destination place) for the permission to send the mobile agent.
6. The foreign place keeper verifies if reception of the mobile agent is possible (safety and probably restricted access). In the positive case it queries also its Agency Keeper.
7. If the answer is positive the Place Keeper registers the mobile agent as a coming mobile agent in a limit time out, generates a password that it sends to the departure place keeper.
8. At the reception of a positive answer the Place Keeper forwards the received message to the agent.
9. The mobile agent stores its password, the list of persistent roles and asks the Site to move him.
10. The Site agent serializes the mobile agent creates and launches the mirror agent, establishes a connection with the distant Site Agent and sends the mobile agent.
11. The agent is generated in the new agency; it has a new current address. It informs mirror agent.

5. THE MOBILE AGENT SCRIPT LANGUAGE (MASL)

5.1 Mobility: Existing Types and Proposed Solution

We distinguish in literature essentially two main kinds of mobility the strong mobility and the weak one. The solution we propose can be situated in a middle level between the two main types. In the following sections we present this solution and how the use of the Itinerary algebra offers a new vision of agents' movements that we adopted for the MASL language. The main types of mobility are:

5.1.1 Strong Mobility

The code, the data and the state of the mobile agent are transferred when it migrates. Even the execution stack is transferred at the moment of migration. Once the mobile agent arrived in the new host, it resumes its activity exactly where it was stopped before the migration operation. Although it is a powerful concept, it is very hard to implement. In fact, this requires the management of threads, and the restoration of their Flow State which is very hard to do. Some approaches tried to modify the Java Virtual Machine (JVM). For instance, the work of (Bouchenak & al 2002) tried to resolve the thread maintenance problem. The idea was to provide an extension of the JVM (jdk1.2) in order to offer an easy manipulation of threads (capture of their state, serialization and restoration). But this extension wasn't adopted for the next versions of JDK.

5.1.2 Weak Mobility

Only the code and the data of the agents are transferred (Robert, S. G., 1995). The agent should restart its activity from either the initial state or from break points programmed by the developer. This mechanism is easy to implement, and the JVM offers the most important primitives to do it (serialization, Java RMI…). The inconvenient is that, the developer has to program, these break points unless the agent should restart its execution from the beginning. Moreover by putting the break point inside the code of the mobile agent means that a high level concept which is mobility is treated at a very low level by designer. Developer should know nothing about how practically the migration is done. They focus only on resolving application problems.

5.1.3 Proposed Mobility

By studying the existing mobility languages two major theories emerge: The languages based on the pi-calculus and those based on the ambient algebra. Although these two theories are very interesting and offer a lot of useful concepts for presenting mobility, we focused on another issue that corresponds best to our vision

of mobile agents. In fact we consider that a mobile agent is created for doing some tasks and has to move from its birth agency in order to accomplish these tasks. Mobile agent's activity should be considered as a mission, once achieved agent returns back. This vision is very similar to the itinerary algebra philosophy (Loke S.W. and al 2003) for which an itinerary describes which actions the mobile agents should execute, where and when. Developers should focus on the applications' problems rather than mobility mechanisms, it is important to offer primitives with a minimum work. A weak mobility with mechanisms, where developers will be brought to insert break points in the code of their agents in order to avoid starting execution from their initial points after migration, is not a suit solution. We propose a mechanism based on implicit break points thanks to the MASL script language. For each migration operation of its itinerary, the agent will be considered in a break point, its activity will restart at this point and a new operation could be executed. Each migration is a new cycle of operations that will end at the departure of the agent to another agency. The designer should conceive and develop mobile agents from the itinerary point of view. Break points are implicit, and are programmed indirectly thanks to the script language components. The advantage of this solution is that the gap between the conceptual and development phase is much reduced. In fact, mobile agents are seen as entities executing an itinerary at the conception level, and during the development phase the developer concentrates its activity on writing the itinerary code of agents based on the MASL syntax.

5.2 The MASL language

MASL combines elements from the agent oriented programming languages (for representing agents' actions) with inspiration from the concurrent language (itinerary algebra) for representing agents' mobility. MASL is structured hierarchically. The main component is the mission that is a set of operations, each operation contains actions. One action is the association of a migration and commands. The syntax of MASL is:

Mission: Id, Name, delay, Operation+	Command: Name, Arguments*
Operation: Id, Name, delay, Action+	Action: (Migration|NULL), Id, Name, delay, command*

Now, get ready for a detailed tour of the concepts and patterns of the MASL language.

5.2.1 Mission

Mission is a global goal that the agent has to achieve. This goal could be assigned by a human. We consider a mission as a set of operations. A mission is identified by an id, a name, contains a list of operations and a time out. The treatment of a mission failure will depend on the designer needs. If there is no mission to execute the mobile agent is in a waiting state and behaves as a stationary agent.

5.2.1 Operation

An operation is a sub goal of a mission. It is a set of actions. Using the operation concept allows the designer to make the problems to resolve fewer complexes. Each mission can be composed of different operations depending on the way how the designer interprets the problem. This flexibility of modelling the problem is one of the major advantages of our language. The way to build missions from a set of operations is a very important phase and designer should take into account all the possibilities in order to optimize the agent's mission. An operation contains a list of actions. As for missions, operations can have also time out.

5.2.2 Action

An action is the association of a migration and a set of commands. It is the concept that the designers can use to order the agent to move. An operation can require a lot of treatments in distinct agencies; each treatment is called an action. In each operation if the agent has to move in another agency, it has to make this in a form of an action in the MASL language. As for missions and operations an action is identified by a number that specify its range in the list of the operation it belongs to. A name and a delay can be associated to an action.

5.2.3 Command

This is the finest element of the MASL language in term of granularity. A command is simply a method in the JAVA language that should be programmed in the mobile agent code. A command has a name that corresponds to the method in the agent java class. The command contains a list of arguments that are the parameters of the method in the Java class. The developer should program his agents' methods keeping in mind that some of these methods will be used in the script language.

6. TRAVEL EXEMPLE

The overall architecture for the application consists of a mobile agent having access to different distant agencies and acts on behalf of a human user. Its responsibility is to search a low cost plane ticket from Paris to Tunis. The interactions between agents and the user are performed via an xml file containing the itinerary following the MASL language. The design and implementation was more focused on validating the requirements than fancy features. The application is composed of two separate parts: the code of the mobile agent, the script describing its itinerary and activity. A third part is the code of place and agency keepers of the application that will not be detailed in this work. The application scenario implemented is described below. We assume three companies: TunisAir, AirFrance and NouvelAir, where a Madkit mobile agent platform is deployed for each one in a country. We suppose that each city is represented by a Place in the agency representing the company in a country. When a customer wants to buy a ticket, a mobile agent, representing him, migrates to different agencies in order to buy the lowest price ticket satisfying user's choices. The birth agency is called the ClientAgency, the agent belongs to the TravelPlace.

6.1 Code of the Mobile Agent

The abstract MobileAgent class inherits from Agent class of Madkit. Every mobile agent is an extension of the MobileAgent and has to implement the method negociateMigration. This method can be used to allow sophisticated acceptance mechanisms of mobile agents in places or agencies. The user specifies the maximum price he can pay for a ticket (250) and the month of its travel (July). A mobile agent's life cycle is based on the execution of missions. The MobileAgent class provide the method ("executeMission") that takes a Mission as parameter and executes it (all the operation actions and commands). The list of missions is extracted from the script written by the developer in the xml file. The method used to extract these missions is provided by the MobileAgent class ("createItinerary"). It takes the path of the xml file as parameter.

```
package madkit.mobility.demo;
import madkit.kernel.*;
import madkit.mobility.*;
public class Migrator extends MobileAgent{
   int nbMig=0, price=250, etat=0;
   public String month="July";
   public Migrator(){
          super("Market");
try{path+=" \\madkit\\mobility\\demo\\mission.xml";
          createItinerary(path);
}
catch(Exception e){}            }
public void negociatePrices(int price, String month){
      println("Ticket : price of "+price+" in "+month);
      // Code of negociation …}
public void live(){
      while(true){
       exitImmediatlyOnKill();
       if(listMissions != null && listMissions.size()>0){
            try{
executeMission((Mission)listMissions.elementAt(0));
            }
            catch(Exception e){}
            }
            pause(10000);}
            }
public void handleMessage(Message m){}
public void negociateMigration(MigrationAnswer answ){ }
public void ShowResults(MigrationAnswer answer){ }
public void end(){}}
```

6.2 Itinerary of the Mobile Agent

For the example we treat we consider that there is a single mission that consists on "searching a low cost ticket". To this end the migratory mobile agent has to move to all these places, negotiates prices and come back. The script language is enough flexible to make it possible to solve this problem with other methods. The first one is to consider moving to search for a low cost ticket in a place and returning back to birth agency in order to show the results as an operation and then do the same thing for the other places. The mission is so divided in three operations. The second way is to consider searching for the best offer of a ticket in the three places as an operation and returning back to show the results to the user as a second one. In this case the mission is divided in only two operations. The first itinerary is:

(**Mission** (Name lowestPriceTravel) (**Operation** (Name negociatePrices)
(**Action** (MoveToPlace TunisAir Paris) (Name fromTunisair) (**Cmd** (Name negociatePrices) (Args 250 July)))
(**Action** (MoveToPlace AirFrance Paris) (Name fromAirFrance) (**Cmd** (Name negociatePrices) (Args 250 July)))
(**Action** (MoveToPlace NouvelAir Paris) (Name fromNouvelAir) (**Cmd** (Name negociatePrices) (Args 250 July))))
(**Operation** (Name show) (**Action** (MoveToPlace clientAgency clientPlace) (Name showPrices) (**Cmd** (show))))))

7. CONCLUSION

In the introduction, we have claimed that the AGR model can not support mobility aspect and that MAGR model is the extension that can resolve this problem. The proposed model is based on the MASIF specification for mobile agent technology. We have presented the AGR model and explained why it can not support mobility. We also presented the MASIF vision of mobility and explained that the problem of MASIF is that it doesn't specify how communications are organized between agents, there is no social conceptual solution and proposed organization for mobile agents. As for AGR the MAGR architecture does not describe the "how", and only specifies the "what" by describing organizational structures made of group, place structures and roles (persistent and non persistent ones). In addition to groups, the MAGR provides a way for partitioning a system through the concepts of agencies and places. Thus, the main lack of AGR disappears: it is possible to organizationally host mobile agents. By designing AgencyKeeper and PlaceKeepe which are specialized agents, we prevent unauthorized and unsecured mobile agents to join agencies and places. Mobile agents, met in different places in the network, can play some specific roles in these places even if they migrate to other agencies; this is possible thanks to the notion of persistent roles. We explained how technically we offered this solution using the MirrorAgent a static agent replacing the mobile one in its birth agency when the latter is in migration. We have defined MASL, a script language, to represent mobile agents' activities. This script language is inspired from the itinerary algebra. In this language mobile agents are driven by missions and all their life cycle is a list of missions that are subdivided on a list of operations. We explained how the use of this script language avoids developers and designers from putting recovery points. Using our language makes the MAGR mobile model nearest to the strong mobility rather than to the weak one. The organizational concepts we present may be used for practical implementations. The MadKit platform that we have extended offers a mobility service that is built around the MAGR model. Since mobile agent technology open environment (P2P networks, Grid, Internet) is relatively new, there is a great deal of research ahead of us. As an initial next step, we plan to build the mobile agent service on the JXTA Platform in lieu of using standard (IP Address: Port mechanisms). Thus, mobile agents will be accessible by all applications even in very dynamic environment. Moreover security should be more robust. This will permit us to easily test scalability, and to continue to explore the model in industrial applications.

REFERENCES

Aglets http://www.trl.ibm.co.jp/aglets

Bouchenak. S., Hagimont. D.,2002., *Services de mobilité et de persistance des applications Java.* Chapitre 6 du livre : Les intergiciels développements récents dans CORBA, Java RMI et les agents mobiles, Hermès, 2002.

David, K. and Robert, S.G., 1999. Mobile Agents and the Future of the Internet. ACM Operating Systems Review, *33(3).*

ElFallah.S.A 2003 CLAIM:Un langage de programmation pour des agents autonomes, intelligents et mobiles. JFSMA'03

Ferber. J, Gutknecht. O., 1998., Aalaadin: a meta-model for the analysis and design of organizations in multi-agent systems, *Ed., Third International Conference on Multi-Agent Systems, Paris, IEEE, 1998.*

Gutknecht O. Ferber. J., 2000.,Madkit:A generic multi-agent platform, 4th Conference on Autonomous Agents

Gerald. K and al, 2002. Path-based Security for Mobile Agents. *Electronic Notes in Theoretical Computer Science 2002*

Harrison, C. G. D. M. Chess, and A. Kershen-baum., 1995. *Technical report, IBM Research Report*

James, E. White., 1997. pp. 437-473: Mobile Agents. *Publisher MIT Press Cambridge, USA*

Loke, S.W. and Zaslavsky, A., 2003. Mobile Agent Supported Cooperative Work: The ITAG scripting language approach. *In Agent Supported Cooperative Work.* pp 205 - 229, Kluwer Academic Publishers.

MASIF: The OMG Mobile Agent System Interoperability Facility. *www.**omg**.org/*

Robert, S. G., 1998 *Agent Tcl:* A flexible and secure mobile-agent system. *Technical Report PCS-TR98-327, 1998.*

Michael S.and al, 1998. Mobile Agents and Security. *In IEEE Communications Magazine 1998.*

Multiagent Systems, Artificial Societies, and Simulated Organizations, Vol 15 pp 95-122. Publishers Springer

Poggi.A,. &al, Modeling Deployment and Mobility Issues in Multiagent Systems Using AUML, Agent-Oriented Software Engineering IV2004, Volume 2935/2003, pp 69-84.

Rafael H. Bordini, and al., 2005 *Claim and Sympa: A Programming Environment for Intelligent and Mobile Agents.* In

White.J., 1996 *Telescript Technology: Mobile Agents.*General Magic White Paper