Towards an MDE Methodology to Develop Multi-Agents Systems including Mobile Agents

Tahar Gherbi[1], Isabelle Borne[1] and Djamel Meslati[2]
[1]*IRISA Laboratory, South Brittany University, Vannes, France*
[2]*Department of Computing, University of Annaba, Annaba, Algeria*

Keywords: Mobile Agent, MAS, MaSE, M-GAIA, AALAADIN, PIM, MDE.

Abstract: There is a need for agent oriented software engineering methodologies that support the conceptual modeling of mobile-agents systems. For this reason, we have presented in a previous work, our meta-model to design multi-agents systems including mobile agents and we have discussed it versus some formalisms extending UML for mobile-agents modeling. The proposed meta-model serves as a platform independent meta-model in our model-driven engineering approach under elaboration as a methodology for the development of multi-agents systems including mobile-agents. This paper summarizes the different approaches for mobile-agent modeling and situates our meta-model particularly versus three works supporting mobility by extending a multi-agents systems methodology (MaSE, GAIA, and AALAADIN). It aims to justify the choices that have guided our meta-model construction.

1 INTRODUCTION

Mobile agents are a promising paradigm for the design and implementation of distributed applications. They have known considerable enthusiasm in the research community, although they have not been translated into a significant number of real-world applications.

Research on mobile agents has been underway for over a decade, particularly in the areas of network management and electronic commerce. Then with, among others, the rapid development of wireless networks, the spread of mobile devices using networks, the development of new networks (such as the Wireless Sensor Networks) and the innovation in the field of Cloud Computing, there was an increase in the use of mobile agents. Applications based on mobile agents are being developed in industry, government and academia; and experts predict that mobile agents will be used in many Internet applications in the coming years (Rajguru et al., 2012).

A mobile agent is a software agent that can, during its execution, move from one site to another, to access data and/or resources. It moves with its own code and data, but possibly with its execution state also. The agent decides independently about its movements. Therefore, mobility is controlled by the application itself and not by the runtime system as is the case of processes migration in operating systems.

There are no specific applications for mobile agents (Milojicic, 1999). In fact, mobile agents are likely to complete or replace the traditional paradigms of client-server architecture, such as message passing, remote procedure call, remote object invocation and remote evaluation. Thus, any application made with mobile agents can be made with any traditional paradigm. The use of mobile agents is, however, advantageous in heterogeneous and dynamic environments that are the trend of modern Internet applications (Cao et al., 2012). Indeed, mobility is of great interest for applications whose performance varies depending on the availability and quality of services and resources, as well as the volume of data moved over network links subject to long delays or disconnections; running on ad hoc networks, or including mobile devices.

However, mobility is not an interaction as an agent does not need to be mobile to communicate. This motivated the inclusion of the mobility model in the design phase (Sutandiyo et al., 2004). Indeed, the development of mobile-agents applications was generally done without considering the mobility aspect in the analysis and design phases. It was often treated in the implementation phase (Belloni et al., 2004). Including this aspect in the analysis and

Gherbi T., Borne I. and Meslati D..
Towards an MDE Methodology to Develop Multi-Agents Systems including Mobile Agents.
DOI: 10.5220/0004420200450055
In *Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering* (ENASE-2013), pages 45-55
ISBN: 978-989-8565-62-4

design phases allow for a better design of this kind of applications: it gives to the designer the ability to use mobility to fulfil the goals of his mobile-agents application (Self et al., 2003).

For this reason, we have presented in (Gherbi et al., 2012), our meta-model for the design of MAS (Multi-Agents Systems) including mobile agents and we have discussed it versus some formalisms extending UML (Unified Modeling Language) for mobile agents modeling. In this paper we summarize, in section 2, the different approaches for mobile-agents modeling. In section 3, we situate our meta-model versus particularly three works extending MAS methodologies to support mobility: (Self et al., 2003) extending MaSE (Multiagent Systems Engineering), (Sutandiyo et al., 2004) extending GAIA and (Mansour et al., 2007a) extending the AGR (Agent, Group and Role) meta-model of AALAADIN, which is a part of our meta-model. The goal is to justify the choices that have guided our meta-model construction. Section 4 presents a case study and section 5 concludes the paper and evokes future work.

2 RELATED WORKS

According to (Loukil et al., 2006), mobile-agents applications modeling can be done by three approaches: design patterns approaches, as in (Aridor et al., 1998; Lima et al., 2004), formal approaches, as in (Picco et al., 1999), and semi formal approaches, in which we distinguish two classes (Bahri, 2010) : formalisms extending UML notations, as in (Belloni et al., 2004; Da Silva et al., 2005; Kusek et al., 2005; Loukil et al., 2006)[1], and approaches extending a MAS methodology, as in (Self et al., 2003; Sutandiyo et al., 2004; Mansour et al., 2007a).

Weary of inventing and re-inventing solutions to recurrent problems, agent design patterns can help by capturing solutions to common problems in agent design (Aridor et al., 1998). However, design patterns have fields of action which are more or less restricted and need to be known. In addition, most of the mobile-agent design patterns presented in literature are difficult to apply in practice due to the lack of a suitable approach to identify, document and apply them (Lima et al., 2004). Formal approaches are good in formalizing simple systems, but for large systems a visual notation is needed to easily grasp the specifications and to specify the system from different points of views. Therefore, we were interested in semi-formal approaches.

Most of the works on semi-formal approaches propose formalisms extending UML. Some address only one aspect of mobility, such as the mobility path, as in (Kusek et al., 2005); some fix the set of sites where the agent can move, as in (Belloni et al., 2004); some include details from MASIF (Mobile Agent System Interoperability Facility), as in (Belloni et al., 2004), or from FIPA[2] (Foundation for Intelligent Physical Agents) standard for interaction, as in (Da Silva et al., 2005). (Belloni et al., 2004) suggest to work more on methodological aspects, by exploring how an existing software development process can be extended to incorporate notations. They recommend the exploration of the Unified Process which seems to be the most appropriate. These formalisms are useful, good contributions and sources of inspiration. However, to contribute in bridging the gap between AOSE (Agent Oriented Software Engineering) methodologies and mobile-agent systems, as suggested in (Milojicic, 1999) and realized in (Self et al., 2003; Sutandiyo et al., 2004; Mansour et al., 2007a), we were interested to extend a MAS methodology. Merging these two areas provides more capacity to solve complex problems in distributed computing becoming increasingly mobile (Self et al., 2003).

Only few works on semi-formal approaches extend a MAS methodology to support mobility. We have encountered three in literature (Self et al., 2003; Sutandiyo et al., 2004; Mansour et al., 2007a). (Self et al., 2003) have extended the MaSE methodology. Figure 1 presents a graphical overview of MaSE which consists of two phases and several steps. The progression over steps occurs with outputs from one step becoming inputs for the next. The result of the MaSE analysis phase is a set of roles that agents will play, a set of tasks that define the behavior of specific roles, and a set of coordination protocols between those roles. The design phase models consist of agent classes, communications defined between them and components that comprise them. Typically, tasks

[1] Other formalisms were discussed in (Gherbi et al., 2012).

[2] FIPA proposed a set of specification with main emphasis on higher level issues like communication language, while OMG (Object Management Group) focused on mobile agents. Since, the two organizations worked independently without any coordination, the end result was the evolution of two parallel standards i.e. FIPA and MASIF. These standards provide specifications and guidelines to developers of frameworks in constructing any agent framework.

from the analysis phase are transformed into components in the design phase. These, possibly multiple, components define the internal agent architecture for each agent defined by the designer.

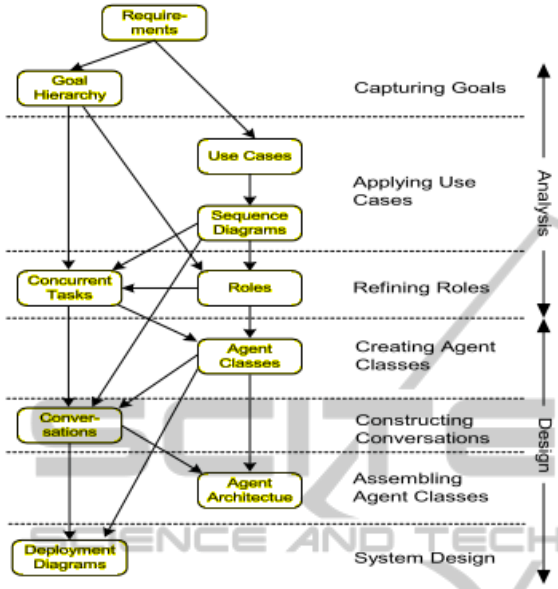

Figure 1: MaSE methodology (Self et al., 2003).

To support mobility, Self et al. have added in the analysis phase a *move* command (to use it in *Concurrent Task Diagrams* describing the behaviors of *Concurrent Tasks*), and in the design phase, *mobile components* that allow the specification of the activities that result from the *move* command. Consequently, an agent is composed of components which are stationary or mobile (a mobile component contains at least one *move* activity). To control and coordinate these components, each agent contains an *Agent Component*, which fulfils also much of the agent mobility functions.

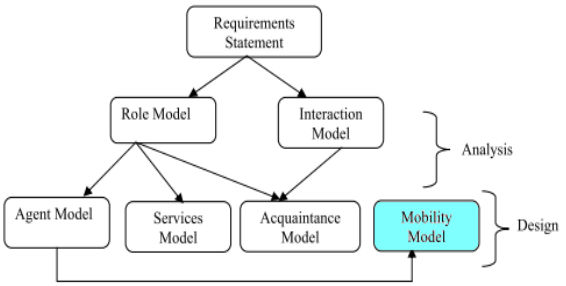

Figure 2: Structure of m-GAIA's models (Sutandiyo et al., 2004).

(Sutandiyo et al., 2004) have criticized the extended MaSE as it does not distinguish conceptually between mobile and stationary agents (even if it does it at the components' level), and because it extends the object-oriented approach rather than starting with a "pure" multi-agents background. They have proposed (figure 2) m-GAIA (mobile GAIA), which distinguishes between mobile and stationary agents in the *Agent model* and defines three role types (system, interface and user) in the *Role Model*. In addition, a *mobility model* was added; it manages concepts of *place types* (locations), *atomic movement* (the smallest granularity movement required to accomplish the task assigned) and *travel path* (a combination of atomic movements). Agent's moves occur at the end of *atomic movements*.

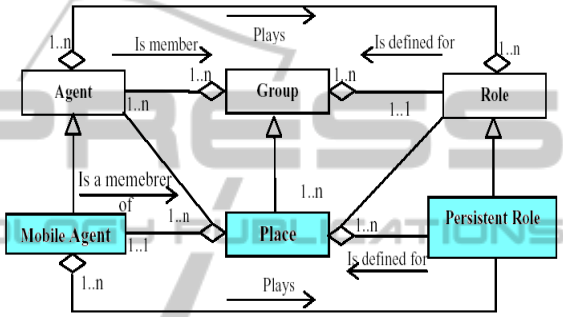

Figure 3: MAGR meta-model (Mansour et al., 2007a).

(Mansour et al., 2007a) note that the existing meta-models and methodologies do not provide any organizational solution for designing and administrating mobile agents in an agent society, and propose MAGR (Mobile AGR) to support the agent's mobility at the organizational level. MAGR enriches the AGR (Agent, Group, Role) meta-model with concepts of *place*, *mobile agent* and *persistent role* (figure 3). A *place* (MASIF concept) represents in MAGR a group joined by only mobile agents; it proposes to them necessary services to move and perform actions. Agents join groups to play roles. When a mobile agent plays a role, it specifies if it is persistent or not. When it moves, all skills associated to a persistent role remain available; however, it will be automatically deleted from any list of agents playing a non-persistent role in the place.

In the presence of mobility, the MAGR's meta-model deals with the social aspect of the agent's life cycle. This is not the case with m-GAIA and the extended MaSE: when an agent moves nothing is done at organizational level. Indeed the role concept is not used after the analysis phase in both methodologies. In addition, social aspects (group, organization) are not clearly defined in MaSE, unlike organizational rules or conversations; and the

developed architectures are static[3]. Similarly in Gaia, the organization and services offered by the agents are clearly static in time, as there is no hierarchical presentation. (Bernon et al., 2009)

Finally and according to (Amor et al., 2004; Jarraya, 2006), MDE (Model Driven Engineering) helps in bringing the gap between MAS's methodologies (as the majority does not include the implementation phase[4]) and platforms[5]. However, we have not encountered an approach based on MDE and extending a MAS methodology to support mobility. Indeed, MaSE uses RUP (Rational Unified Process), m-GAIA uses the cascade model and MAGR does not propose an elaborated process[6]. Therefore, our goal is to propose an MDE methodology to develop mobile-agents applications.

The choice of MDE is justified also by its benefits (know-how durability, productivity grain and heterogeneous platforms consideration), which explain its adoption in many works on various fields, including MAS, as in AMDD for INGENIAS (Pavon et al., 2005), MDAD (Jarraya et al., 2007), ASPECS (Cossentino et al., 2009) and ASEME (Spanoudakis et al., 2010). In addition using MDE may facilitate the mobile agent moves across heterogeneous platforms: rather than sending the agent's code, we send its model which can be transformed into code on target sites.

3 CHOICES THAT HAVE GUIDED OUR META-MODEL CONSTRUCTION

Choosing a MAS methodology is difficult (Amor et al., 2004; Jarraya, 2006). In the absence of a consensus on a meta-model to design MAS (despite the unification efforts of well-known MAS meta-models, as in (Cossentino et al., 2005; Beydoun et al., 2009)), we have looked for a meta-model which is simple to use, modular and evolutive, in order to extend it and supports agents mobility.

Our choice fell on the PIM (Platform Independent Model) meta-model of MDAD (Model Driven Agent Development) for several reasons. Firstly, it is based on the AEIO decomposition (from the VOYELLES approach (Demazeau, 2001)) which considers a MAS as composed of four bricks (or vowels A,E,I,O)[7]: Agent, Environment, Interaction and Organization. This provides modularity at the models' level, rather than at the level of agents and agent's skills. The ability to interchange and reuse models of each brick has a strong potential for reuse and versatility, as there is no presupposition to use a particular model a priori (Jarraya et al., 2007). Secondly, its organizational meta-model, based on AGR, does not imposes constraints about the internal architecture of the agent, its behavior, or its capabilities. Thirdly, MDAD is already a model driven methodology illustration for the stationary-agents applications development.

Inspired from the related works, we have enriched its PIM meta-model with the stereotypes (figure 4): «MobileAgent», «Site», «Migration» (to prepare the agent before calling the Jump Action), «Jump» (to move effectively the agent to another site), «Clone» and «AfterMigration» (to integrate correctly the agent in the MAS, after its move to a new site).

The concepts in gray boxes, the two associations between «SendMessage» and «ReceiveMessage» (added to ease code generation (Gherbi et al., 2012)), the *transfearable* tagged-value in the «DomainConcept» stereotype, and the *stop* tagged-value in the «Role» stereotype are those we have added. According to figure 4, a group contains several roles and an agent (which may be stationary or mobile) may play several roles. However to play a role, the agent must join the group containing this role, and then ask for authorisation.

We assume that the agent determines when it is necessary to move. However, other agents, or the agent platform itself, may advise the agent to move (for example, for shutdown, load balancing, etc.); in this case, the agent's autonomous nature allows it to determine whether it will actually move (section 4 gives some guidelines to help treating this case). We also assume that the agent platform handles the effective move of agents: when it receives an agent's

[3] O-MaSE (Organization-based MaSE), an extended version of MaSE (DeLoach, 2005), defines a meta-model for agents to adapt their organization during execution.

[4] Meta-models in GAIA and AGR are generic: i.e. they make abstraction on the internal architecture and the behavior of agents. The passage to the implementation phase of such methodologies remains informal and manual. (Jarraya, 2006)

[5] MAS methodologies and platforms generally represent multi-agents concepts differently. (Jarraya, 2006)

[6] AGR can be seen as complementary to other agents centered methodologies, because it is insufficient alone to represent all aspects of multi-agents (Jarraya, 2006). Indeed MAGR (as AGR) does not provide meta-models for agents, roles and domain.

[7] A fifth vowel (U for User) has been added in (Demazeau, 2003).

move request (generated from the «Jump» action), it terminates the agent and sends it to the destination platform where it is restored.

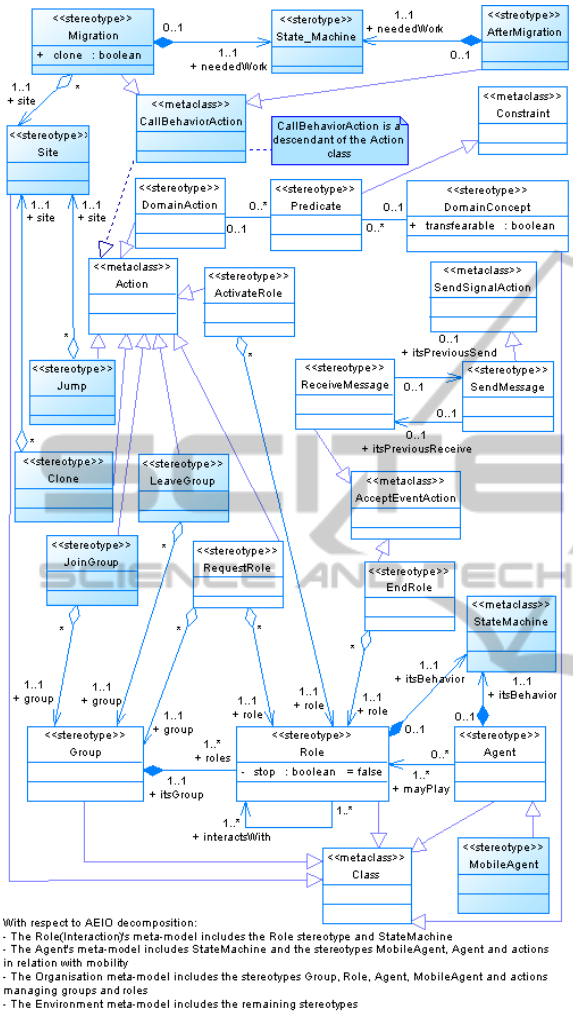

Figure 4: A PIM meta-model for MAS including mobile agents.

Unlike MDAD, agents and roles goals are not expressed explicitly, but implicitly via theirs behaviors (they can also be noted as comments). However, if an explicit expression is needed, one can use for example OCL (Object Constraint Language) constraints as in MDAD (figure 5).

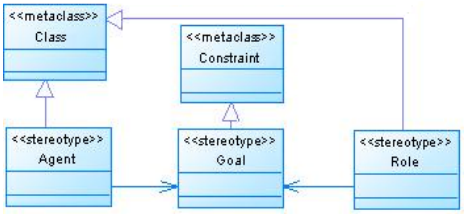

Figure 5: Goals modelling in MDAD.

In another hand, unlike MDAD, we describe behaviors with state-charts diagrams, as in (Self et al., 2003; Loukil et al., 2006), to save transformation effort (because we will use state-charts diagrams to model behaviors at the PSM level also)[8].

Compared to the published version in (Gherbi et al., 2012), we have added the «MobileAgent» stereotype to distinguish between stationary and mobile agents and have a direct mapping from PIMs to PSMs (Platform Specific Model) of mobile-agents platforms: Indeed, if some mobile-agent platforms, like JavAct, do not make this distinction, others like Grasshopper, do. We have also added an association between «Clone» and «Site» stereotypes to allow flexible cloning independently of migration. The clone concept, which importance was mentioned in (Self et al., 2003), was not modeled in the extended MaSE, m-GAIA and MAGR. Finally, we have added a stop tagged-value (with *false* as default value) in the «Role» stereotype to be able (when an agent want to move) to end roles held in parallel (see the case study).

In some related works, the mobile-agent itinerary is modeled to capture its movements' path, as in (Belloni et al., 2004), or to describe its mission by defining tasks to do on each site of the itinerary, as in (Sutandiyo et al., 2004; Da Silva et al., 2005; Loukil et al., 2006). We do not model this, because mobile-agents platforms normally maintain information on agents movements path, which can be requested; and for the agent's mission, it is described via its behavior.

We also do not fix the set of sites where a mobile agent can move, as in (Belloni et al., 2004): we assume that agents are intelligent enough to sense their environment and discover sites where they may (if necessary) move. Otherwise, the model may become unreadable in presence of lot of sites; in addition, sites are not usually all known for all applications at the design phase (e.g. in ad-hoc networks).

Finally, we encourage local communications between agents and so we support only non-persistent roles. Consequently before leaving a site, a mobile agent must release all held roles, as in (Da Silva et al., 2005). The persistent roles of MAGR generate distant communications: indeed, queries for a service provided by a persistent role will be relayed to a mirror agent representing the mobile

[8] To model behaviors, MDAD uses, at PIM level, activity diagrams and, at PSM level, ATN (Augmented Transition Network); thus, it defines transformation rules between them.

agent playing this role. Knowing that one of the mobility goals is to reduce the network traffic, is it really efficient for a requesting agent to see its requests relayed to a mirror agent residing on a remote site (the mobile-agent native site) rather that interacting with the concerned mobile agent by sending messages directly to it or by moving up to it?

Our meta-model serves as a PIM meta-model for an MDE approach which is under elaboration as a methodology to develop MAS including mobile-agents. Figure 6 shows its steps. We have elaborated a PSM meta-model for JavAct (a mobile-agent platform), represented the PIM and PSM meta-models with respect to Ecore format (using Eclipse/EMF and UML2Profiles), and defined the transformation rules from PIM to PSM, as well as, the code generation rules from PSM to JavAct's code. The parts which remain under development are: automation of transformations (using ATL: Atlas Transformation Language) and code generation (especially, from stereotyped state-charts diagrams).

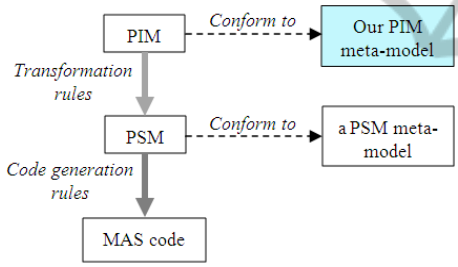

Figure 6: an MDE development process[9] for MAS.

4 CASE STUDY

Consider (figure 7) a simple library database distributed on site1, site2 and site3. On each site, a stationary agent (*Librarian*) deliver the list of all books stored locally. Using a laptop, we create on site1 a mobile agent (*MobileBookSeeker*) to search for the locations of a given book over a given itinerary (e.g. site1, site2 and site3); then the laptop can disconnect. The mobile agent visits all sites, asks on each one for the local books list and filters it to check if it contains the searched book. When it finishes, it moves to its final destination (the laptop when it is connected) to deliver its results. Using mobile agents is obviously advantageous in this case.

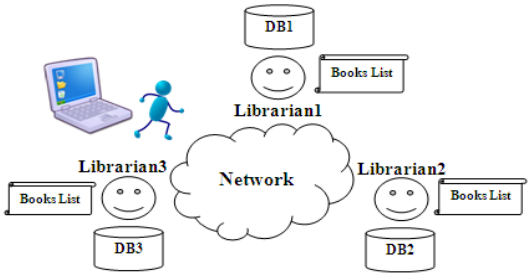

Figure 7: A book searcher application example.

A PIM for this example is given in figure 8. The *LibraryManagement* group contains three roles. The *Librarian* agent plays the *BooksListDeliver* role; and the *MobileBookSeeker* agent plays, on each visited site, the *BookChecker* role which interacts with the *BooksListDeliver* role to get the local books list. When the *MobileBookSeeker* finishes its mission, it plays the *ResultsDeliver* role to deliver the list of repositories of the searched book.

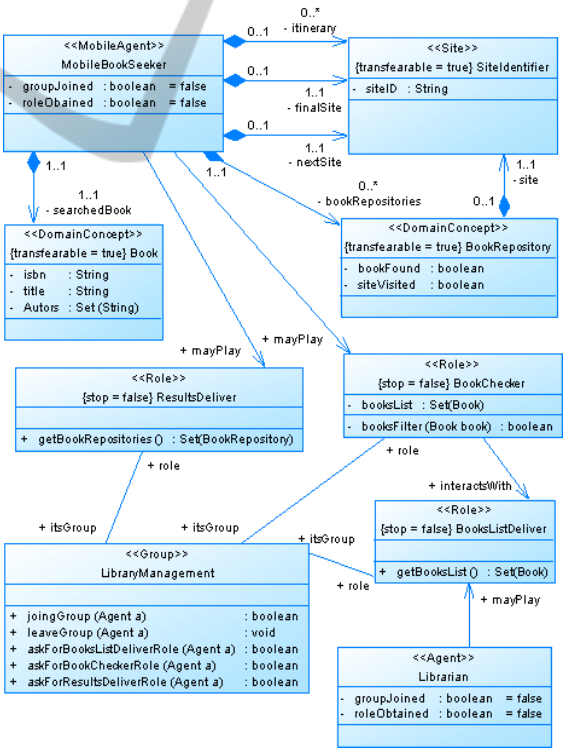

Figure 8: The classes diagram for the application example.

Each agent (or role) has an attribute *itsBehavior* (not presented in figure 8 for a better readability) pointing to the state-chart describing (in a separate figure) the agent (or role) behavior.

[9] MDAD has not proposed a CIM (Computation Independent Model) meta-model.

The behaviors of the *Librarian* agent and the *BooksListDeliver* role are given in figures 9 and 10 respectively.

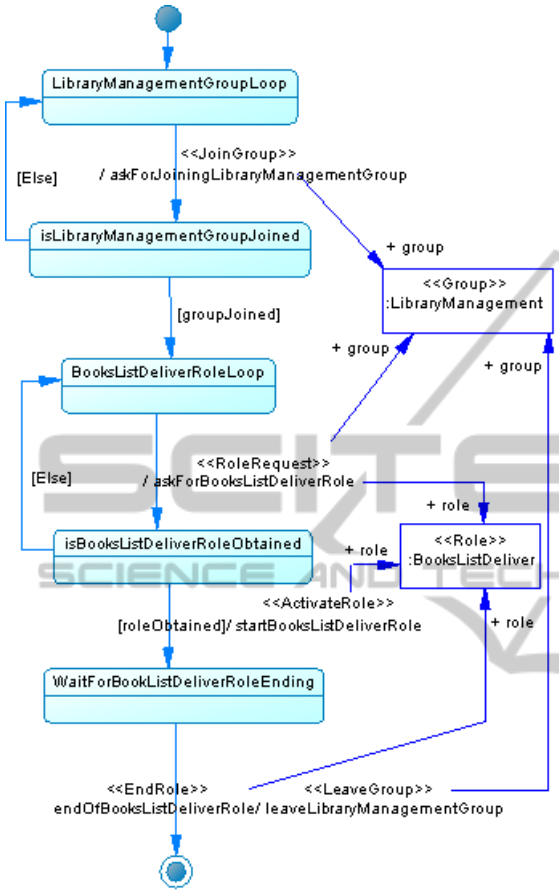

Figure 9: Librarian behaviour.

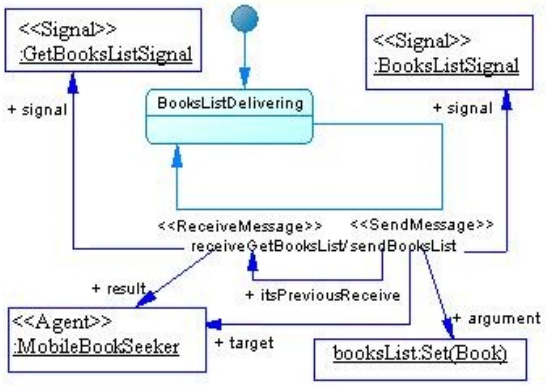

Figure 10: BooksListDeliver behaviour.

The *Librarian* (figure 9) joins the *LibraryManagement* group, asks to play the *BooksListDeliver* role and leaves the group when the role ends. When playing the *BooksListDeliver* role (figure 10), it waits unlimitedly for requests to deliver its local books list.

The behaviors of the *MobileBookSeeker* agent, the *BookChecker* role, and the *ResultsDeliver* role are given in figures 11, 12 and 13 respectively.

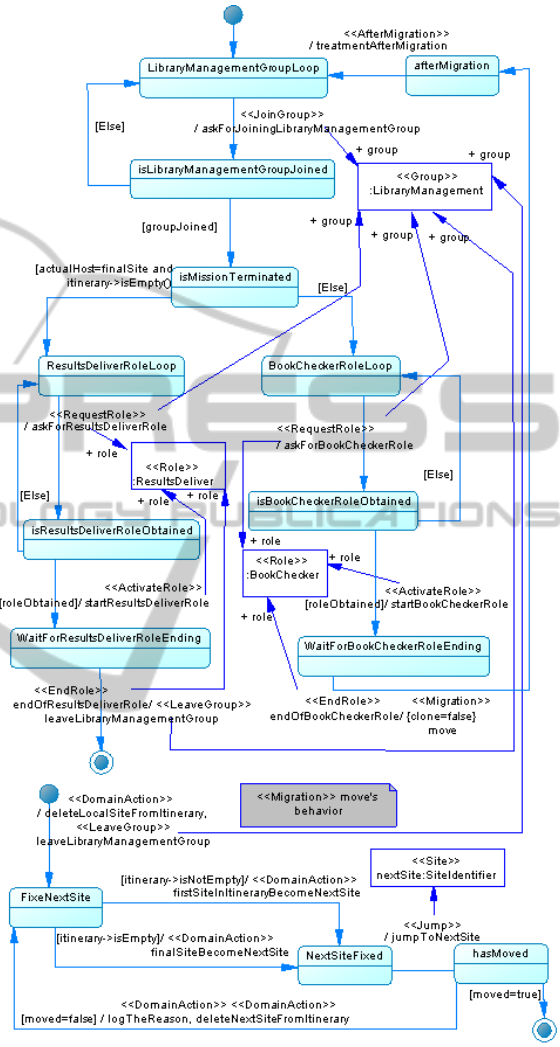

Figure 11: MobileBookSeeker behavior.

The *MobileBookSeeker* agent joins the *LibraryManagement* group (figure 11), then checks if its mission is terminated. If yes, it plays the *ResultsDeliver* role and leaves the group when the role ends; else, it plays the *BookChecker* role and then moves to the next site in the itinerary.

Migration and *AfterMigration* actions have their own behaviors (state-chart), where the designer may include actions which he judges necessary. For our example, the *Migration* action leaves the group, determines the next site, and jumps to it; where the *AfterMigration* action does nothing.

Migration and *AfterMigration* actions may become complex, for example if a mobile agent playing roles in parallel is needed. The agent may inside the *Migration* action ask the currently held roles to stop, wait for them to end, note from the stopped services (furnished by these roles) those it judges necessary for its activity after the move, and leaves the groups of held roles. Inside the *AfterMigration* action, the agent may search, as described in (Mansour et al., 2007b), for roles furnishing the noted services, joins their groups and plays them.

To stop a role, its *stop* tagged-value must be made to *true*; and inside its behavior, this attribute must be checked to know if the role can continue or if it must stop and end.

Moves requested by an external entity (another agent or the agent platform), can be considered, for example, by adding an *externalMoveRequest* tagged-value in the «Agent» stereotype (with *False* as default value). Thus an external entity can request an agent to move by setting its tagged-value to *True*. When entering in any state (in its state-chart diagram representing its behavior), the agent checks this tagged-value: if it is *True*, its saves the name of the current state[10] and launches the *Migration* action. Tthe *AfterMigration* action terminates by passing the agent into the saved state.

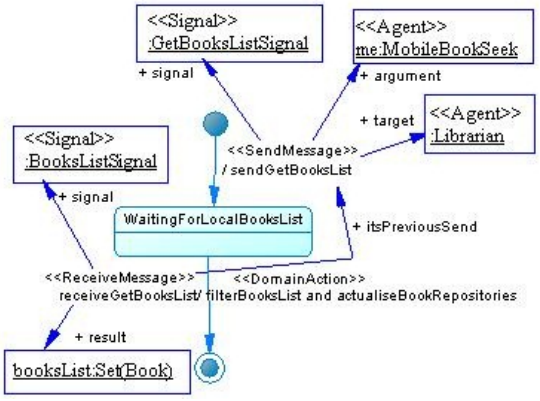

Figure 12: BookChecker behaviour.

When playing the *BookChecker* role (figure 12), the agent sends a *sendGetBooksList* message, waits to receive the list, then checks if it contains the searched book. When playing the *ResultsDeliver* role (figure 13), the agent waits until it deliver its results.

[10] Or the name of the next state if the current state is to wait for the end of a role (i.e. if its name has the form WaitForrolenameRoleEnding).

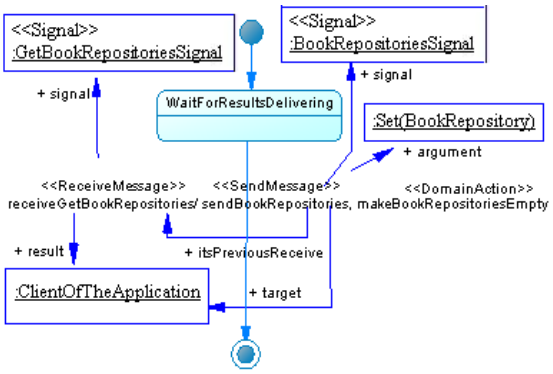

Figure 13: ResultsDeliver behaviour.

In sections 3 and 4, we have discussed the similarities and differences between our proposed meta-model and the studied works. To see this in practise, let us model the same example using the studied methodologies. We recall that we interest only to the mobility modelling.

Using the extended MaSE, the modelling of our example, produces the agent classes in figure 14 (showing the roles played by agents), and the roles diagram in figure 15 (showing the association between roles and the concurrent tasks *searchBook*, *deliverResults*, and *deliverBooksList*).

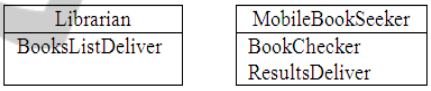

Figure 14: Agent classes.

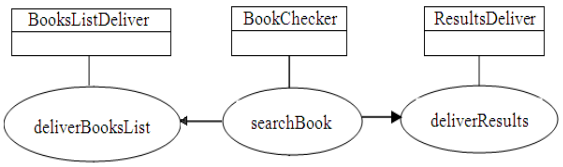

Figure 15: Roles diagram.

Bellow, we present only the concurrent task diagram for the *searchBook* task (figure 16), and its corresponding mobile-component (figure 17).

The task begins (figure 16) by testing if the mission is completed. If yes, it sends a *missionCompleted* message to the *ResultsDeliver* role. Else, it sends a *getBooksList()* message to the *BooksListDeliver* role, waits for the local books list, checks if it contains the searched book (and eventually actualise the repositories list), then tries to move to next site.

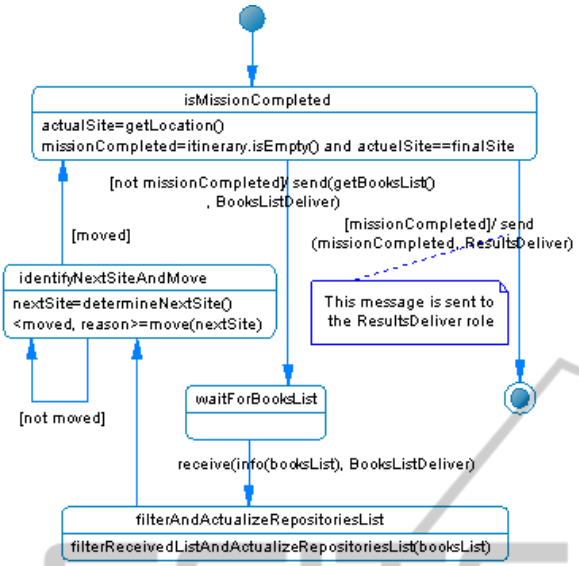

Figure 16: searchBook task.

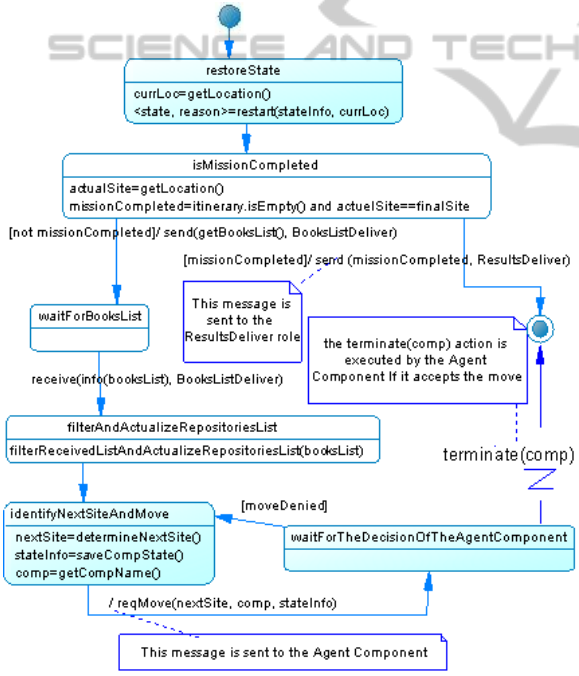

Figure 17: Mobile searchBook component.

In the *identifyNextSiteAndMove* state (figure 17): when a mobile component wants to move, it saves its state, informs its *Agent component* and waits for its decision. If the Agent component refuses, it replies by a *moveDenied* response; else it terminates the mobile component and orders all other components to save their states and send them to it. Every time it receives a state, it terminates the sender component. The *Agent Component* terminates, when all components terminate. Then the agent moves with all components and theirs saved states. At the target site, the *Agent Component* restarts all components and communicates their saved states to them. The *restoreState* state identifies the state in which the component restarts after migration; for the case of the *searchBook* task, the component restarts always in the *isMissionCompleted* state.

Using m-GAIA, we identify in the *agent model* two types of agents: $MobileBookSeeker_m$ and *Librarian*, where the index (m) indicates that the agent is mobile. We also identify the following roles in the *role model*: *BooksListDeliver* (system role), *BookChecker* (interface role) and *ResultsDeliver* (user role). Figure 18 illustrates the relationship between the roles and the agent types.

Figure 18: Agent model for our example in m-GAIA.

In the *mobility model*, we distinguish two types of places: *mobilePlace* (with instance=1, to represent the laptop) and *stationaryPlace* (with instance=3, to respresent site1, site2 and site3). $MobileBookSeeker_m$ can run on the two types of place where *Librarian* can run only on the *stationaryPlace* type. The *mobility model* allows, in addition, the elaboration of the *travel schema* for the mobile agent, which defines its origin place type (*stationaryPlace*: site1 for our example), its destination place type (*mobilePlace*: the laptop for our example) and a set of travel paths (each one is a list of atomic movements). For our example, one travel path suffices. However, details about the syntax of atomic movements were not given in (Sutandiyo et al., 2004): the authors have modelled their application example, realized it separately on Grasshopper, and then made manual correspondence between the modelled example and its realisation.

The MAGR's concepts (except *place* and *persistent role*) are the base of the organization in our proposed meta-model (see figure 4). Thus (organizational) models realized with MAGR are closer to ours. However, MAGR does not propose meta-models for agent, role, and domain. After the elaboration of the organizational model, it passes to the development step where it proposes MASL (Mobile Agent Script Language) to program MAS on Madkit (a mobile agent platform, supporting AGR and MAGR and compliant to MASIF). MASL has a vision which is similar to the itinerary algebra

Table 1: Mobility modeling in MAS methodologies.

	before/after migration's Treatment	Itinerary modeling	Mobile/stationary agent distinction	Considering organizational aspects with mobility	Development process
Extended MaSE	yes by *Agent Component*	no	at level of components	no	RUP
m-GAIA	not needed	yes	yes	no	Cascade
MAGR	not needed	no (and yes at level of implementation)	yes	yes	do not propose an elaborated process (*)
Our PIM proposition	yes	no	yes	yes	MDE[12]
(*) The development cycle is quite limited. Gutknecht and Ferber have never wanted to propose a real process, in order to keep AGR generic and not reduce its potential of integrating into ascendants or descendants processes. (Gauthier, 2004)					

philosophy for which an itinerary describes which actions the mobile agent should execute, where and when (Mansour et al., 2007). With MASL, a mobile agent seems as executing a mission (representing its global goal). A mission is a set of operations (representing sub goals of the mission). An operation is a set of actions (each one is a treatment executed on a different site). An action contains a move instruction and a set of commands (the finest elements of MASL).
The script describing the itinerary and activity of the mobile agent in our example can be elaborated as below[11]:

```
(Mission (Name findBookRepositiories)
(Operation (Name searchBookRepositiories)
    (Action (MoveToPlace Librarian Site1) (Name
    bookChecker) (Cmd (Name getBooksList)) (Cmd
    (Name booksFilter) (Args searchedBook)))

    (Action (MoveToPlace Librarian Site2) (Name
    bookChecker) (Cmd (Name getBooksList)) (Cmd
    (Name booksFilter) (Args searchedBook)))

    (Action (MoveToPlace Librarian Site3) (Name
    bookChecker) (Cmd (Name getBooksList)) (Cmd
    (Name booksFilter) (Args searchedBook)))
)
(Operation (Name deliverBookRepositiories)
     (Action (MoveToPlace clientAgency Laptop) (Name
     resultsDeliver) (Cmd (deliverBookRepositories)))
)
)
```

As shown, only MAGR and our meta-model consider organizational aspects (group, role) in the presence of mobility.
On another hand, m-GAIA and MAGR support the agent mobility by structuring its behavior as an itinerary which describes the task to do on each site; consequently, no effort is needed before or after moving. In contrast, the exended MaSE (respectively, our meta-model) allows for more flexibility in modeling the agent's behavior, and employs a *move* action (respectively, *Migration* action); however, an effort is needed before moving to save the states of the agent's components (respectively, to release roles and leave groups), and after moving to restore components (respectively, to eventually join groups and obtain roles).

Table 1 summarizes the discussion between methodologies extending MAS to support mobility. It interests only to the question of modelling mobility in the presented methodologies; for a comparison between MAS methodologies on others criteria see, for example, section 2.5 in (Bernon et al., 2009), section 6 in (Cossentino et al., 2009) and section 6 in (DeLoach et al., 2010).

5 CONCLUSIONS

The complexity and scope of software systems continue to grow. One approach to deal with this growing complexity is to use intelligent MAS (DeLoach et al., 2010).

This paper contributes to bridge the gap between AOSE methodologies and mobile-agent systems, as our proposed PIM meta-model serves to develop MAS including mobile agents. In (Gherbi et al., 2012), we have situated our meta-model versus some formalism extending UML notations. In this paper, we have summarized the different approaches to model mobile-agents and particularly three works extending MAS methodologies (MaSE, GAIA, and AALAADIN) to support mobility; we have situated our meta-model versus them and have discussed the choices that have guided its elaboration.

Our meta-model was slightly updated, compared to its published version in (Gherbi et al., 2012), to distinguish between mobile and stationary agents, to support flexible cloning and to treat, inside the *AfterMigration* action, the case when a mobile agent

[11] Codes of mobile-agent, place and agency keepers are not shown.

[12] For details on MDA/MDE, see (Gherbi et al., 2009).

wants to move while holding (and eventually playing) roles.

As a future work, we will first illustrate our MDE approach by transforming the PIM example built here into a PSM for JavAct, then into JavAct code. We will also discuss the issue of mobile-agents platforms compliance with MASIF and FIPA specifications. After, it will be necessary to conduct experiments with real applications using different mobile-gents platforms to validate and enrich the proposed approach.

REFERENCES

Amor, M., Fuentes, L., Vallecillo, A., 2004. Bridging the Gap Between Agent-Oriented Design and Implementation Using MDA. *In AOSE, New York, pp. 93–108.*

Aridor, Y., Lange, D. B., 1998. Agent design patterns: elements of agent application design. *In AGENTS'98, USA, pp. 108-115.*

Bahri, M. R., 2010. Une approche intégrée Mobile-UML/Réseaux de Pétri pour l'analyse des systèmes distribués à base d'agents mobiles. *Doctoral thesis, University of Constantine, Algeria*.

Belloni, E., Marcos, C., 2004. MAM-UML: an UML profile for the modeling of mobile-agent applications. *In the 24th SCCC, Arica, Chile, pp. 3-13.*

Bernon, C., Gleizes, M.-P., Gauthier, P., 2009. Méthodes orientées agent et multi-agent. *Technologies des systèmes multi-agents et applications industrielles. chapter 2*. A. El Fallah-Seghrouchni; J.-P. Briot (Ed.).

Beydoun, G., Low, G., Henderson-Sellers, B., Mouratidis, H., Gomez-Sanz, J. J., Pavon, J., Gonzalez-Perez, C., 2009. FAML: A Generic Metamodel for MAS Development. *In Journal of IEEE Transactions on Software Engineering, Vol. 35(6), USA, pp. 841-863.*

Cao, J., Das, S. K., 2012. *Mobile Agents in Networking and Distributed Computing.* Wiley Series in Agent Technology, John Wiley & Sons, Inc., USA.

Cossentino, M., Bernon, C., Pavon, J., 2005. Modelling and meta-modelling issues in agent oriented software engineering. *The AgentLink AOSE TFG.*

Cossentino, M., Gaud, N., Hilaire, V., Galland, S., Koukam, A., 2009. ASPECS: an Agent-oriented Software Process for Engineering Complex Systems, How to design agent societies under a holonic perspective. *In AAMAS, Vol. 20(2), pp.260–304.*

Da Silva, V. T., R. Noya, C., De Lucena, C. J. P., 2005. Using the UML 2.0 Activity Diagram to Model Agent Plans and Actions. *In AAMAS'05, pp. 594-600.*

DeLoach, S. A., 2005. Engineering Organization-Based Multiagent Systems. *SELMAS, USA, pp. 109-125.*

DeLoach, S., A., Garcia-Ojeda, J., C., 2010. O-MaSE: a customisable approach to designing and building complex, adaptive multi-agent systems. *Int. Journal of AOSE, Vol. 4(3), pp. 244-280.*

Demazeau, Y., 2001. VOYELLES, *HDR (Habilitation to Direct Research) thesis, INP Grenoble, France.*

Demazeau Y., 2003. Créativité émergente centrée utilisateur. *In Briot J., Khaled G. (dir.), Déploiement des systèmes multi-agents – Vers un passage à l'échelle. JFSMA'03, Hermès - Lavoisier (Reviue RSTI Hors-série). pp. 31-36.*

Gauthier, P., 2004. Méthodologie de développement de systèmes multi-agents adaptatifs et conception de logiciels à fonctionnalité émergente. *Doctoral thesis, University of Paul Sabatier, France.*

Gherbi, T., Meslati, D., Borne, I., 2009. MDE between Promises and Challenges. *In the 11th Int. Conf., Comp. Modeling & Simulation, UKSim'09, Cambridge. pp. 152-155.*

Gherbi, T., Borne, I., Meslati, D., 2012. Un méta-modèle pour les applications basées sur les agents mobiles. *In CIEL-2012, Rennes, France, pp. 1–6.*

Jarraya, T., 2006. Réutilisation des protocoles d'interaction et démarche orientée modèles pour le développement multi-agents. *Doctoral thesis, University of Reims, France.*

Jarraya, T., Guessoum, Z., 2007. Towards a model driven process for multi-agent system. *5th Int. CEEMAS,Vol. 4696, Leipzig, Germany. pp. 256-265.*

Kusek M., Jezic, G., 2005. Modeling Agent Mobility with UML Sequence Diagram. *In AOSE, Ljubljana (Slovenia), pp. 51-63.*

Loukil, A., Hachicha, H., Ghedira, K., 2006. A proposed Approach to Model and to Implement Mobile Agents. *In IJCSNS, Vol. 6(3B), pp. 125-129.*

Lima, E. F. A., Machado, P. D., Sampaio, F. R., Figueiredo, J. A., 2004. An approach to modeling and applying mobile agent design patterns. *In ACM SIGSOFT, pp. 1-8.*

Mansour, S., Ferber, J., 2007a. MAGR: Integrating mobility of agents with organizations. *In IADIS, Portugal.*

Mansour, S., Ferber, J., 2007b. Un modèle organisationnel pour les systèmes ouverts déployés à grande échelle. *In JFSMA'07, Carcassonne, France, pp. 107-116.*

Milojicic, D., 1999. Mobile agent applications (trend wars). *In IEEE Concurrency, Vol. 7(3), pp. 80-90.*

Pavon, J., Sanz, J. G., Fuentest, R., 2005. The INGENIAS Methodology and Tools. *In Agent-Oriented Methodologies, B. Henderson-Sellers and P. Giorgini, eds., pp. 236-276.*

Picco, G. P., Murphy, A. L., Roman, G. C., 1999. Lime: Linda Meets Mobility. *In ICSE'99, pp. 368 – 377.*

Rajguru. P. V., Deshmukh. S. B., 2012. Current trends and analysis of mobile agent application. *In proceedings of NCETCT-2012, WJST, Vol. 2 (3), India, pp. 1-6.*

Self, A., DeLoach, S. A., 2003. Designing and Specifying Mobility within the Multiagent Systems Engineering Methodology. *In 18th ACM SAC, USA, pp. 50-55.*

Spanoudakis, N., Moraitis, P., 2010. Using ASEME methodology for model-driven agent systems development. *In AOSE conf., Toronto, pp. 106-127.*

Sutandiyo, W., Chetri, M. B., Loke, S. W., Krishnaswamy, S., 2004. Extending the Gaia Methodology to Model Mobile Agent Systems. *In ICEIS, Porto, pp. 515-518.*