MAMT: an environment for modeling and implementing mobile agents

Héla HACHICHA
SOIE
ISIMS, Institut Supérieur d'Informatique et de Multimédia, Sfax
hela.hachicha@fsegs.rnu.tn

Adlèn LOUKIL
SOIE
INSAT, Institut National des Sciences Appliquées et de Technologie, Tunis
adlen.loukil@insat.rnu.tn

Khaled GHEDIRA
SOIE
ENSI, Ecole Nationale des Sciences de l'informatique, Tunis
khaled.ghedira@isg.rnu.tn

ABSTRACT
This paper presents an approach to model and to implement mobile agents. This approach is materialized by a UML profile, called MA-UML for modeling mobile agents, and a software development environment that assists the specification, design and implementation stages of the agent system development lifecycle, called MAMT. The MAMT environment provides support for modeling multi-agent systems by using the MA-UML profile.

Keywords
Mobile Agent, UML, AUML, UML profile, mobile agent engineering.

1. INTRODUCTION
Mobile agents are software entities that can migrate autonomously throughout a network from host to host. This means they are not bounded to the platform they begin execution. Mobile Agents are emerging as an alternative programming-concept for the development of distributed applications.

So far, most of the work on the area of mobile agents has been focusing on the technology itself, and the development of agent frameworks to support mobility. However, few works have proposed to model mobile agent-based application and no formalism yet exists to sufficiently specify mobile agents.

In this context and in order to contribute towards to solve this problem, we have been working to propose an approach to model and to implement mobile agents. This approach is materialized first by the MA-UML (Mobile Agent UML) profile, which extends the UML language [1] and the AUML formalism [2]. Second, the proposed approach provides also the MAMT (Mobile Agent Modeling Tool) software CASE Tool, which supports the use of MA-UML profile and the generation of Java code from conceptual diagrams in order to implement mobile agent-based applications.

This paper is structured as follows. Section 2 describes the considerations to model mobile agents and reviews the previous approaches to model mobile-agent applications. In section 3, we describe an overview of the MA-UML profile. Section 4 shows the MAMT environment, describing its architecture and main features. Then, it presents our strategy for mapping conceptual specifications to Java code. Finally section 5 summarizes the paper and offers directions for future work.

Jung, Michel, Ricci & Petta (eds.): *AT2AI-6 Working Notes, From Agent Theory to Agent Implementation, 6th Int. Workshop,* May 13, 2008, AAMAS 2008, Estoril, Portugal, EU.

2. MOBILE AGENTS MODELING
2.1 Design considerations
We discuss in this section the basic concepts of mobile agent needed for its specification. According to the literature [3], the mobility of an agent is related to some concepts such as: itinerary, location, move action, remote cloning action, and security. In addition, a Mobile Agent (MA) must contain all of the following models: an agent model, a lifecycle model, a computational model, a security model, a communication model and a navigation model. Also MA must exist in a software environment (called, Mobile Agent Environment) in which it can execute.

Based on these issues, we have identified the following concepts to consider when modeling mobile agent:
Concept 1: the environment that describes entities of a mobile-agent application.
Concept 2: the internal structure which describes MA characteristics.
Concept 3: the itinerary which describes the list of locations to visit or to reside and the set of tasks to be performed in order to complete a specific mission.
Concept 4: the travel schema which describes the travel planning.
Concept 5: the tasks execution planning which describes the mapping of tasks to be performed on different locations.
Concept 6: the interactions which describe the communication acts.
Concept 7: the security mechanisms which describe the security properties needed to protect agent from malicious entities and to protect entities from malicious mobile agents.
Concept 8: the lifecycle model which describes MA's behavior.
Concept 9: the move action (weak or strong) and the cloning action.
Concept 10: the mobility paths which describes the different network nodes.

2.2 Mobile Agent modeling with UML
The literature defines three main categories of approaches to model mobile agents which are: the design pattern approach, the formal approach, and the semi-formal approach.

The semi-formal formalisms can be classified into two classes. The first class is the semi-formal approaches that propose their

own methodologies or extend their agent-oriented methodologies such as: the MaSE methodology [4] was extended to allow the analysis and the design of mobile agent. The m-Gaia [5] proposed extension of the Gaia Agent Oriented software Engineering (AOSE) methodology to model mobile agent systems. The second class of approaches aims to propose some extensions to UML or\and AUML notations.

In our work we are interested to the semi-formal approaches and particularly to formalisms which proposed extensions to UML or\and AUML notations. We mention hereafter some of them and the most relevant for our work.

The MAM-UML [6] profile has introduced new stereotyped classes, packages, relationships, and different tagged values to model entities of a mobile-agent application and their structural relationships. Additionally, it has proposed to describe the itinerary model using the UML interaction diagram (sequence or collaboration). In this diagram, authors have defined new stereotyped messages, using the dependency relationship (move, remote cloning), and constraints which associated to the mobility messages enable to specify why and when an agent moves. Mobile agent's behavior during its lifecycle has modeled using the UML statechart diagram and by defining new stereotyped actions (strong move, weak move, and remote cloning).

Moreover, a variant of an UML activity diagram is used to specify model of activities. Different swimlanes break the diagram into different types of places and different constraints associated to a transition between activities enable to specify why and when an agent moves. Interactions of mobile agents are modeled using the UML collaboration diagram as a template. Finally, the mobility view is modeled using the deployment and component diagrams.

The advantage of the MAM-UML approach is the covering of the most mobile agent concepts in the three phases of the development process (analyze, design and implementation). But, some limits can be identified in this work. In fact, the MAM-UML itinerary model describes the agent's mobility between locations and the interactions of MA with entities (stationary agents and resources). However, in the literature [7] "the itinerary typically consists of a list of tasks to be executed sequentially and the locations where the tasks are to be performed".

Also the travel planning designed in itinerary model is static which not considers the environment changes that can occurs and make agent incapable to reach some locations (e.g. the mobile agent platform on the destination address is not operational, the machine that the agent is moving from is isolated from the rest of the network). Moreover, the tasks execution planning designed with UML activity diagram is not dynamic (with a static set of locations). Finally, authors have not considered the design of mobile agent internal structure, security mechanisms, and mobility path.

Klein et al. [8] have proposed extensions to UML class diagram with new stereotypes and tagged values to model entities involved in a mobile agent application. In addition, stereotyped actions «move» and «remote execution» are introduced in the UML statechart diagram to show the relationship between agent state change and agent location change. Moreover, authors have proposed to specify agent mobility by extending the UML sequence diagram with the new elements and new stereotyped messages (move, remote execution, and clone).

This work represents an initial approach on modeling mobile agent and not considered the most mobile agent concepts.

Mouratidis et al. [9] have introduced extensions to the UML deployment and activity diagrams. These extensions have been integrated to the AUML formalism. The AUML deployment diagram allows developers to specify mobile agent, origin location, destination location, and static aspect of mobility paths. The AUML activity diagram allows developers to specify the dynamic aspect of the mobility path: the sequence of the movement, the detailed mobility path, and the decisions that drive the choice of particular intermediate nodes. The activity nodes model plan, while the transitions model events.

This work has the advantage of the modeling of the mobility paths of agent which considers the environment changes and the decisions which MA will take during its travel. Also, it allows specifying the travel planning of MA. However, this represents a static planning. Moreover, this approach not considers the other concepts of MA.

Kang et al. [10] have proposed extensions to activity diagram in UML 1.5 and UML 2.0 in order to model the dynamic behavior of mobile agents. Authors have introduced the new stereotype «host» with a parameter for a swimlane, which represents the location with a unique name (address). The activity "Go" is also introduced to model agent's movements. Authors have also modeled the exception handling which can occur and cause the failing to access to the destinations hosts.

This extended UML activity diagram allows to specify the tasks execution planning of mobile agent in different hosts. However, this represents a static planning. This work also has not considered the other mobile agent concepts.

Kusek et al. [11] have introduced extensions to UML sequence diagram to model agent mobility, current location, and location of agent creation. Four variants uses of sequence diagram are proposed to model agent mobility. In these diagrams different stereotypes are added: stereotype «agent» to model agent, stereotype «at» to model current location, and stereotype «move» to model agent's movements.

In these diagrams, vertical lines represent both places and instances of agents and arrows between these lines represent movements of agents between places. This makes diagrams so complex and particularly with an important number of places.

2.3 Discussion

All these approaches previously described are useful and interesting contributions. However, no formalism yet exists to sufficiently specify the basic concepts of MA described in section 2.1. Table 1 summarizes the principal concepts of mobile agent and the contributions of some existing approaches.

Table 1. Contributions of some existing approaches

	MAM-UML [6]	Mobile UML [8]	AUML [9]	Kang et al. [10]	Kusek et al. [11]
Environment	×	×			
MA Internal structure					
Itinerary	×	×			
Travel schema	×	×	×		×
Task planning	×			×	
Interaction	×		×		×
Lifecycle	×	×	×		
MA security					
Mobility Path			×		

After examining the presented approaches, some deficiencies can be identified to model some concepts; we summarize some limits in the following points:

The internal structure of MA is not well specified by these approaches.

The security properties needed for mobility is not addressed.

The travel planning of mobile agent modeled by some of these approaches [8] [11] [6] [9] is not flexible and not dynamic (with a static set of locations). In fact, if MA cannot reach some locations, then their related tasks cannot be performed. A mobile agent travel planning is considered one of the most important techniques for completing a given task efficiently. However, a static planning may not be the best approach in real network environments. So it is necessary to model a flexible and dynamic travel planning, which considers the environment changes.

The tasks execution planning modeled by some of these approaches [6] [10], using the activity diagram, is not dynamic (with a static set of locations). In fact, if MA cannot reach some locations for ever reason, then the list of tasks relevant to these locations cannot be performed and MA cannot reach its mission. So it is necessary to model a dynamic tasks execution planning.

In order to contribute towards to surmount some insufficiencies, we have been working to propose additional extensions to UML and AUML notations. This represents the MA-UML profile, presented in the next section.

3. THE PROPOSED MA-UML PROFILE

It is to be notice that our works focus on the modeling of mobile agent applications when there are only two kinds of entities: mobile agents and static locations; and they are not well suited for mobile computing modeling (laptops, mobile phones, PDAs).

There are seven diagrams in MA-UML profile classified into static and dynamic diagrams. MA-UML extends the UML and the AUML class diagrams and defines three new diagrams to model mobile agent structural / static aspects, which are environment diagram, mobile agent diagram and itinerary diagram. MA-UML extends the UML statechart diagram, the UML activity diagram, and the AUML sequence diagram to model mobile agent dynamic aspects, which are: lifecycle diagram, mobile agent activity diagram, mobile agent sequence diagram and navigation diagram. Figure 1 illustrates the relationships between these diagrams.

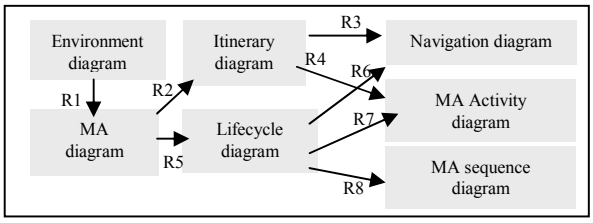

Figure 1. Inter-diagrams Relationships.

3.1 Environment Diagram

The purpose of this diagram is to model all entities involved in a mobile-agent application, their properties and their structural relationships. Typically, this application involves several agents (stationary and mobile) that interact and communicate, playing different roles. Also, it involves one or several region(s), a number of mobile-agent systems, some places, and several resources.

In order to model these entities and their structural relationships, we propose to introduce new stereotyped UML classes («place», «resource»), stereotyped AUML classes («mobile agent», «stationary agent»), stereotyped packages («region», «m-agent-system»), stereotyped associations («communicate», «reside», «manipulate», «home»), and new graphical notations. This extended diagram is called *Environment Diagram*. To illustrate these extensions, we present as an example the environment diagram designed for electronic commerce application (figure2).

Figure 2 illustrates an environment diagram for a simple mobile-agent application. In such application, a SearcherAgent (mobile agent) moves between different places provided by two different mobile-agent systems, in the context of a region. The InterfaceAgent (stationary agent) is responsible for the initialization of SearcherAgent with its mission. The SecurityAgent is responsible for authenticating the SercherAgent. The DataBase represents the resource manipulated by mobile agent.

After modeling the different entities, each mobile agent specified in this diagram must be specified in the mobile agent diagram in order to specify its internal structural and its characteristics (R1, Figure 1: Environment diagram → MA diagram).

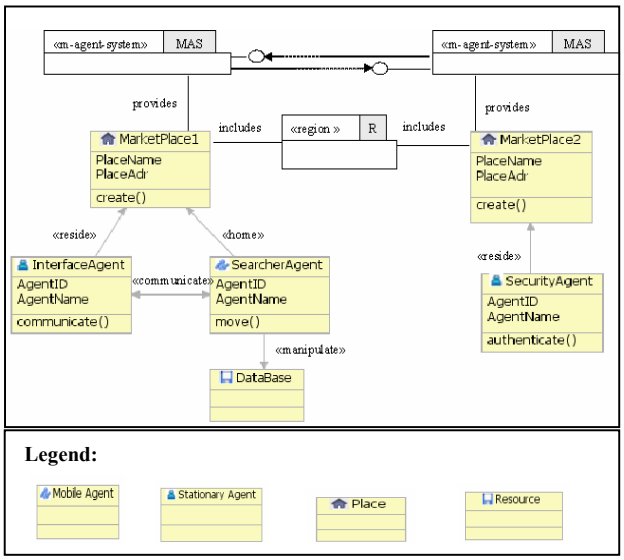

Figure 2. The environment diagram of a mobile-agent application

3.2 Mobile Agent Diagram

This diagram is responsible for modeling the internal structure and the characteristics of a Mobile Agent (MA). The properties of the internal structure of a MA depend on the requirements of applications (electronic commerce, telecommunication, etc.). But we believe that several properties must be identified and are available and needed for each application. Based on the literature [3], we propose to define additional properties in the internal structure of the stationary agent relevant to the mobile agent (e.g. authentication, history, itinerary).

In order to specify the properties relevant to the MA, we propose to extend the AUML agent class diagram by adding these properties as attributes in a separate AUML AgentBaseClass compartment. This extended diagram is called *Mobile Agent Diagram*; figure 3 shows the structure of the extended AUML class diagram.

After specifying the mobile agent internal structure, it is necessary to specify the itinerary diagram (R2, Figure1: mobile agent diagram → Itinerary diagram). Also, for each mobile agent it is necessary to specify the different states that agent can reach during its lifetime (R5, Figure 1: MA diagram → Lifecycle diagram).

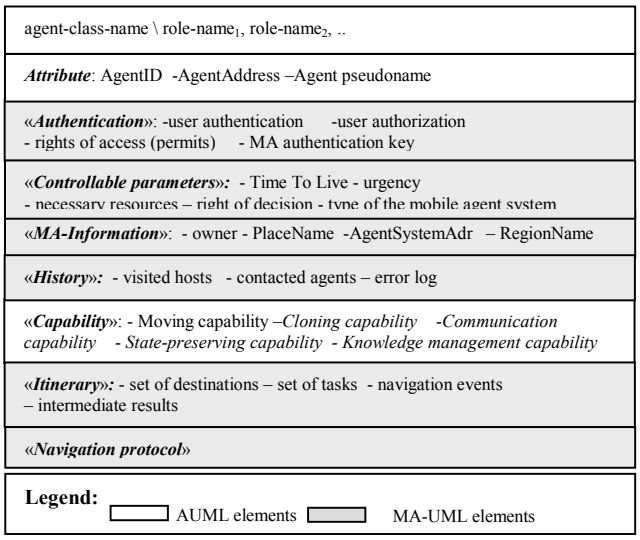

Figure 3. The internal structure of the mobile agent

3.3 Itinerary Diagram

An agent's itinerary describes the tasks of the agent and the locations where those tasks are to be performed [7]. The itinerary model defines the mobile agent travel planning. The travel planning can be determined either statically or dynamically. That is, it can be calculated either before the agent is dispatched or while the agent is migrating. Dynamic travel planning is more flexible, and can adapt to environmental changing in real time. However, since the travel planning is calculated on the fly, it also consumes more computation time and more power of the local sensor. On the other hand, although static travel schema cannot adapt to the network change, it is able to save both computation and power since the travel planning only needs to be calculated once. Computation-efficiency, power-efficiency, and flexibility are three parameters that cannot be satisfied at the same time.

In order to be able to specify a dynamic and flexible travel planning, which can adapt to environment and network changes, we propose that the developer predicts and introduces *Navigation Events* into the itinerary model. The navigation events represent the unexpected events that can be produced during the migration of mobile agent or having learned information from another entity. Moreover, the developer must define additional destinations places (*Equivalent Places*) in the itinerary model. Then, if a navigation event occurs and MA can not reach a given place, it must update its travel planning and move to equivalent place instead of the failure place in order to be able to perform the associated tasks.

In order to model the elements of the mobile agent itinerary model, we define a new diagram, called *Itinerary Diagram*. This diagram represents an extension of the UML class diagram by introducing new stereotyped classes and new graphical notations. Figure 4 describes the elements of the itinerary diagram.

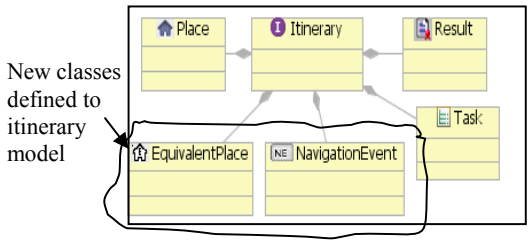

Figure 4. The Itinerary diagram

The set of navigation events that can occur and the equivalent places defined in the itinerary model can be updated by the MA during its travel, having learned any information or having met a problem. Figure 4 presents the static view of mobile agent itinerary model. The dynamic view of itinerary may be viewed as the specifications of the travel planning and the tasks execution planning. In order to model the dynamic view of the itinerary, we define two new diagrams: first the navigation diagram which specifies the travel planning between locations (R3, Figure 1: Itinerary diagram → Navigation diagram). Second, the mobile agent activity diagram which specifies the mapping of tasks to be performed on locations (R4, figure 1: Itinerary diagram → MA activity diagram).

3.4 Mobile Agent Diagram

This diagram is responsible for modeling the tasks execution planning among different places. In UML, activities (tasks) are specified with the activity diagram. In order to model the relationships between locations and tasks and the reuse aspect of an activity, we propose to introduce the concept of location in the UML activity diagram; we call this diagram *Mobile Agent Activity Diagram*. To specify the concept of location, we propose to attach parameters to each activity (parameterized activity). These parameters represent the list of places where this activity (task) needs to be performed. Figure 5 illustrates an example of a mobile agent activity diagram.

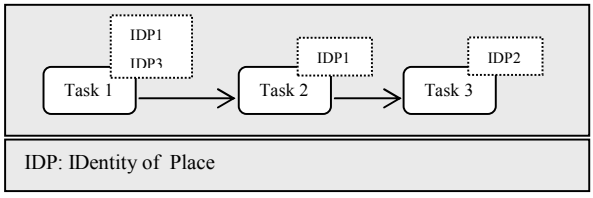

Figure 5. The mobile agent activity diagram

The parameters attached to each task make possible to the developer to specify the set of places (locations) where this task can be performed. In figure 5, task 1 for example is performed in place P1 and place P3. The parameters can be instantiated during the system execution process. That means that the set of places can be updated during the MA execution when it can not reach some places by the equivalent places specified in the itinerary diagram. Then, with the use of parameters it is possible to model a *dynamic tasks execution planning* and to specify the *reuse aspect* of an activity.

3.5 Navigation Diagram

The purpose of this diagram is to model the agent travel planning which describes its movements between places defined in its itinerary.

In order to specify a dynamic travel planning, we propose a variant use of the UML statechart; we call this diagram *Navigation Diagram*. We propose that the states model places, the transitions between states model the movements of an agent between places, and events which trigger the transitions between states model the navigation events which drive the movements of MA to the equivalent places. Figure 6 illustrates an example of the proposed navigation diagram.

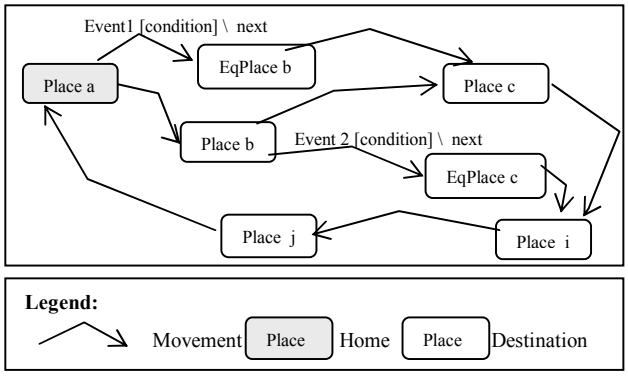

Figure 6. The navigation diagram

Figure 6 shows a navigation diagram that models the travel planning of the agent between different places. In this diagram, when MA is in "place a" and need to move, it passes automatically to the next place ("place b"), but if any navigation event occurs (event1), MA transit to the equivalent place of place b (Eqplace b).

With the use of the UML statechart diagram, it is possible to specify the navigation events that can be occurred during MA travel and to specify additional locations to be visited if a MA can not reach a given location. These issues allow to model a *flexible and dynamic travel planning* which adapt to the environment changes.

3.6 Lifecycle Diagram

The lifecycle diagram extends the UML statechart diagram by adding new stereotyped actions and new types of transitions.

During its lifetime and in order to achieve its mission, MA needs to communicate with other entities (agents, environment, user), to move from location to another, and to perform the assigned tasks. These issues allow to specify when and how mobile agent transits to one state to another. Then, in order to specify mobile agent change states, we believe it is necessary to specify relations with the interaction diagram, which specifies the communication acts, the navigation diagram, which specifies the travel planning, and the activity diagram, which specifies the tasks mapping.

In order to model these relationships, we propose to introduce new stereotyped actions and new types of transitions: activity, interaction, and navigation transitions. The activity transition triggers the execution of the mobile agent activity diagram in the

"Activated" state in order to perform tasks relevant to a given location (R7, Figure 1: Lifecycle diagram → MA Activity diagram). The interaction transition triggers the execution of the mobile agent sequence diagram in a given state in order to execute communication acts (R8, Figure 1: Lifecycle diagram → MA sequence diagram). The navigation transition triggers the execution of the navigation diagram in the "ChooseDest" state in order to determine and to decide to the next location to be visited (R6, Figure 1: Lifecycle diagram→ Navigation diagram). As an example, figure 7 illustrates the graphical notations and the use of navigation transition.

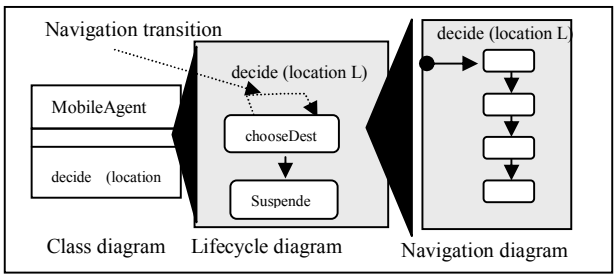

Figure 7. Granularity levels of the lifecycle diagram use: navigation transition

3.7 Mobile Agent Sequence Diagram

The AUML sequence diagram (protocol diagram) [2] presents a set of interactions between agents playing different roles. In order to model the different interactions between the new elements defined by MA-UML profile, we propose to introduce the new proposed elements in the AUML sequence diagram. We call this diagram *Mobile Agent Sequence Diagram*. Table 2 identifies the instances that may appear in the mobile agent sequence diagram and its associated diagram elements.

Table 2. The mobile agent sequence diagram elements

Instances	Mobile Agent	Stationary Agent	Place	Resource
Diagram element	Mobile Agent	Stationary Agent	Place	Resource
Instances example	SearcherAgent	InterfaceAgent	MarketPlace	DataBase

4. MAMT: AN ENVIRONMENT FOR MODELING AND IMPLEMENTING MOBILE AGENTS

In order to support the use of the MA-UML profile and to implement a system using MA-UML, it is necessary to create a software CASE Tool (Computer Aided Software Engineering Tool) and to refine the models and to generate Java code.

In the following sub-sections, we present the software CASE Tool we have developed. Then we describe the proposed strategy for mapping mobile agent specifications to Java code.

4.1 The MAMT environment

We developed MAMT (Mobile Agent Modeling Tool) that is a software development environment to support a mobile agent-based applications development process. The MAMT environment was developed as a set of plug-ins for the Eclipse Platform [12].

The MAMT environment consists of three plug-ins, which are:

The graphical editor tool: includes the MA-UML library. It is composed of the UML metamodel, the AUML metamodel and the MA-UML metamodel. The editor tool supports the seven MA-UML diagrams and allows the creation of a number of UML, AUML and MA-UML artefacts. The GEF plug-in was used to provide a powerful foundation for creating editors for visual editing of arbitrary models. The EMF plug-in was used to create and store UML, AUML and MA-UML models in the XMI format.

The translator tool: includes the transformation rules. It is responsible for the transformation of the MA-UML XMI file, which represents the output of the editor tool, to the UML XMI file.

The code generation tool: includes a Java library and a set of code generation rules. It is responsible to generate automatically Java code based on the UML XMI file. The code should be completed by the developer.

As a MAMT prototype, we have implemented the graphical editor tool using the UML2 2.0, EMF 2.2, GEF, GMF plug-ins of eclipse. This editor supports essentially the three static diagrams of the MA-UML profile which are: the environment diagram, the mobile agent diagram and the itinerary diagram. Also it supports the MA-UML navigation diagram. Figure 8 illustrates as an example the metamodel needed to implement the environment diagram editor.

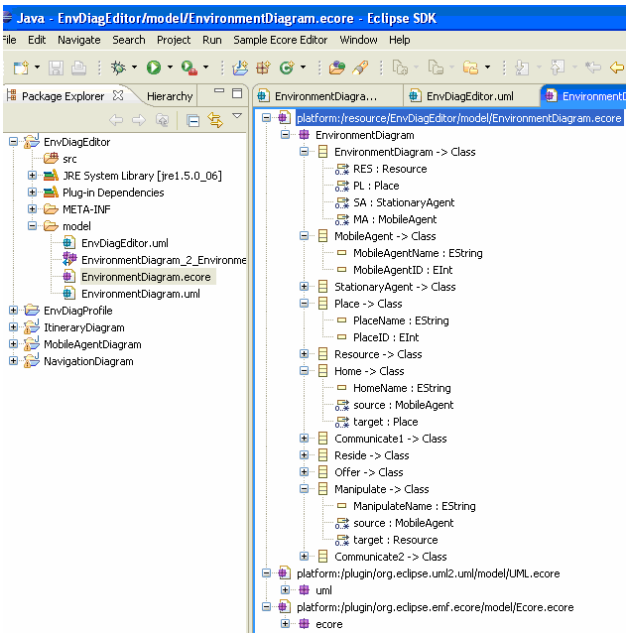

Figure 8. A metamodel part of the environment diagram

Figure 8 shows the metamodel of the environment diagram which describes the different metaclasses added to UML metamodel (e.g. mobile agent, resource, reside). The Editor tool corresponding to the environment diagram is illustrated in figure 9.

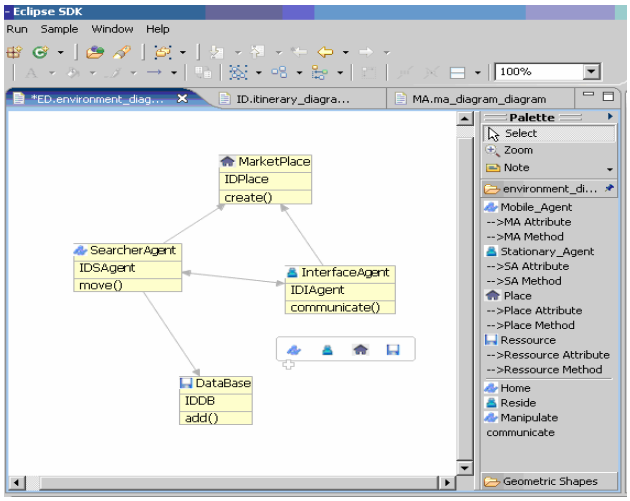

Figure 9. A snapshot of the MAMT tool showing the editing of an environment diagram

4.2 Mapping MA-UML models to Java code

The transformation process of MA-UML models to Java code is supported by the two MAMT plug-ins: the translator tool and the generator tool. In order to map MA-UML models to Java code, we propose to use the MDA approach [13]. The MDA defines a set of consecutive transformations that should be applied to the models in order to allow the transformation of high-level abstraction models into code. The MDA approach proposes four layers allow deriving code from the specifications, which are: CIM, PIM, PSM, and code. The proposed approach for mapping MA-UML specifications to Java code is illustrated in figure 10.

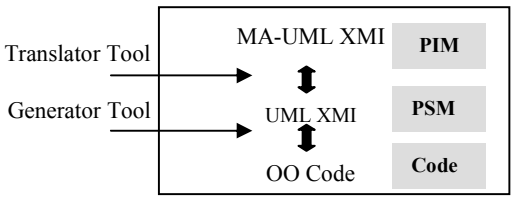

Figure 10. Transformation process based on MDA

The PIM layer. The MA-UML is a modelling language focused to model mobile agent-based application. The MA-UML models that describe an application are PIMs that are portable to diverse systems and can be used to generate different computational models by applying various implementation platforms.

Transforming PIMs into PSMs. After elaborating the MA-UML models of PIMs, these models should be transformed into PSMs. The transformation of MA-UML models into UML models occurs in two stages: (1) the first stage: the first stage consists of describing the all the MA-UML models in a textual description by using the XMI format. The XMI is used in our approach to represent the MA-UML models, to assist in the transformation from MA-UML models into UML models, to represent the UML models and to help the transformation from UML models into code. (2) The second stage: in this stage, the MA-UML XMI generated in the previous stage is converted into a UML XMI. This transformation is based on the transformation rules that we have defined.

The PSM layer. The UML XMI generated in the previous stage represents a PSM of the application. The UML XMI created in this stage is a UML class diagram that contains the classes of the application. All application entities, properties, relationships, and behaviour modelled by the MA-UML diagrams generate the UML class diagram of the application.

Transforming PSMs into code. In the final stage of the transformation process, the UML models are transformed into code. This kind of transformation corresponds to the last stage of the MDA approach that transforms PSMs into code. This transformation is based on the code generation rules that we have defined.

5. CONCLUSION

This paper presented an approach to support the modeling and the implementation of mobile agent. This approach consists first on the MA-UML profile which defines a set of seven diagrams that describes the static and the dynamic aspects of the mobile agent. Second, this approach consists on the MAMT that is a software development environment to support the use of MA-UML diagrams and transform specification models to Java code. The transformation process is based on the MDA approach. Our future works include two axes. In the first, we are looking to implement the other MA-UML dynamic diagrams. In the second axe, we are looking into the design and the implementation of a mobile agent-based application in the medical field.

6. REFERENCES

[1] Unified Modeling Language Specification, version 2.0, OMG, http://www.uml.org. Accessed in: December 10, 2004.

[2] Bauer, B., Müller, J. P., Odell, J. 2001. Agent UML: a Formalism for Specifying Multiagent Software Systems. In International Journal of Software Engineering and Knowledge Engineering, vol. 11, No. 3, pp.1-24, 2001.

[3] Tomoya, T., Tadanori, M., Takashi, W. 1998. A Model of Mobile Agent Services Enhanced for Resource Restrictions and Security. In International Conference on Parallel and Distributed Systems (ICPADS '98).

[4] Self, A., DeLoach, S. A. 2003. Designing and Specifying Mobility within the Multiagent Systems Engineering Methodology. Special Track on Agents, Interactions, Mobility, and Systems (AIMS) at The 18th ACM Symposium on Applied Computing (SAC 2003).

[5] Sutandiyo, W., Chhetri, M. B., Loke, S.W., Krishnaswamy, S. 2004. MGaia: Extending the Gaia Methodology to Model Mobile Agent Systems. In the Sixth International Conference on Enterprise Information Systems (ICEIS 2004), Porto, Portugal, April 14-17.

[6] Belloni, E., Marcos, C. 2003. Modeling of Mobile-Agent Applications with UML. In Proceedings of the Fourth

Argentine Symposium on Software Engineering (ASSE´2003). 32 JAIIO (Jornadas Argentinas de Informática e Investigación Operativa), Buenos Aires, Argentina. September 2003. ISSN 1666-1141, Volume 32.

[7] Ling, S., Loke, S. W. 2001. Verification of Itineraries for Mobile Agent Enabled Interorganizational Workflow. In Proceedings 4th International Workshop on Mobility in Databases and Distributed Systems (MDDS'2001), Munich, Germany, 582-586.

[8] Klein, C., Rausch, A., Sihlinh, M., Wen Z. 2001. Extension of the Unified Modeling Language for mobile agents. In Siau K. Halpin T. (Eds.): Unified Modeling Language. System Analysis, Design and Development Issues, chapter VIII. Idea Group Publishing, 2001.

[9] Mouratidis, H., Odell, J., Manson, G. 2002. Extending the Unified Modeling Language to Model Mobile Agents. Workshop on Agent-oriented methodologies. OOPSLA 2002, Seattle, USA, November (2002).

[10] Kang, M., Taguchi, K. 2004. Modeling Mobile Agent Applications by Extended UML Activity Diagram. In Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS)'04, Porto, Portugal, 519-522, April (2004).

[11] Kusek, M., Jezic G. 2006. Extending UML Sequence Diagrams to Model Agent Mobility. In Agent-Oriented Software Engineering, vol. 4405, pp. 51-63, 2006.

[12] Eclipse: Eclipse.org, v 3.0, http://www.eclipse.org/. Accessed on 05/2005.

[13] OMG MDA Guide. Version 1.0.1, http://www.omg.org/docs/omg/03-06-01.pdf.