# A proposed Approach to Model and to Implement Mobile Agents

*Adlèn Loukil, Héla Hachicha and Khaled Ghedira*

SOIE, Institut Supérieur de Gestion de Tunis, Université de Tunis

**Summary**
In the last years, mobile agents are gaining great attention as a new concept for developing and implementing mobile and distributed applications. However, very little work has taken place in defining concepts and notations to model mobile agents.
In this paper, we present an approach to model and to implement mobile agents. This approach is materialized by a UML notation, called MA-UML for modeling mobile agents, and a software CASE Tool, called MAMT for mapping conceptual diagrams into Java implementation.

***Key words:***
*Mobile Agent, UML, Agent UML, Mobile Agents Engineering*

## Introduction

Mobile Agents (MAs) are a class of software agents that have the ability to move from host to host. As a result of this intrinsic characteristic of mobility, nowadays, mobile agents raise considerable interest as a new programming concept for developing and implementing mobile, distributed and interoperable applications such as Telecommunication Management, Health-Care Medicine, information retrieval, and electronic commerce.

Nowadays there are several technological solutions to support the development of mobile agent-based applications. All of them provide a platform to support the migration and execution of code (agent platform) and use scripting or interpreted languages (such as TCL or Java) to cope with heterogeneity. However, these technologies do not take into account the analysis and the design phases of the development process. 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 by a MA-UML (Mobile Agent UML) notation, which extends the UML [1], and the AUML [2] notations, and the MAMT (Mobile Agent Modelling Tool) software CASE Tool, which supports the use of MA-UML notations and the automatic code generation from conceptual diagrams.

In this paper we present first the MA-UML notation for modelling mobile agents. Then, we describe the MAMT software CASE Tool and our strategy to map the mobile agent specifications to Java code. This code will be integrated into a mobile agent platform in order to implement mobile agent-based applications.

The paper is organized as follows. Section 2 reviews previous approaches extending the standard UML or\and AUML formalism to model mobile agents. In section 3, we describe our approach for modelling mobile agents and we discuss the contributions of the proposed approach. Section 4 presents our strategy and CASE Tool to map mobile agent specifications to Java code. Finally, a conclusion and an outlook to future works are made in section 5.

## 2. Mobile Agent Modeling with UML

In the last years, many agent-oriented modelling techniques and methodologies have been developed, such as Agent UML [2], and GAIA [3]. However, these methodologies, and formalisms are not well suitable to model mobile agents. In this context, in recent years, several approaches (methodologies, formalisms, profiles) for modeling and specifying mobile agents have been proposed. The literature defines three main types of approaches which are: the pattern approach [4], the formal approach [5] and the semi-formal approach [6].

In our work we are interested to 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.

MAM-UML [7] is an approach for the modeling of mobile-agent applications. It aims to propose extensions to the standard UML in order to capture relevant abstractions for the modeling of mobile-agent features. This approach is materialized by an UML profile, which includes views to model organizational, lifecycle, interaction and mobility aspects of mobile-agent applications contributing to the analysis, design and implementation phases of their development.

Klein et al. [8] have proposed some extensions to UML for mobile agents. They have introduced new concepts to model *strong mobility*, *weak mobility* and *cloning action*. Also, they have introduced new stereotypes to model entities of mobile agent environments (mobile-agent, mobile-agent systems, places, and regions). Moreover, they have proposed

extensions to the sequence diagram in order to model agent's movement from one location to another.

Gervais and Muscutariu [9] have defined an Architecture Description Language (ADL) devoted to the design of mobile agent systems. This ADL has proposed some concepts and operation interfaces necessary for interoperability among heterogeneous mobile-agent systems. In addition, it has proposed extensions to deployment and component diagrams and it has introduced new stereotypes to model mobile agents and mobile agent systems. Modeling mobility of agent is supported by the stereotyped flow relationship "*becomes*", an operation "*move*", and operations "*beforeMove*" and "*afterMove*" that prepare the agent for the migration.

Mouratidis et al. [10] have introduced extensions to the UML deployment and activity diagrams to give answers to some questions that arise from the use of mobile agents. These extensions have been integrated to the AUML formalism. The AUML deployment diagram allows developers to capture the reason (*why*) an agent moves to a different nodes and the location (*where*) it moves. The AUML activity diagram allows developers to capture the *when* (timing) a mobile agent leaves a node to move to another.

## 3. The proposed MA-UML notation

### 3.1 The MA-UML notation

According to the literature, the mobility of an agent is related to some concepts such as: *itinerary*, *location*, *move action* (strong or weak), *remote cloning* action, *security*, etc. In addition, based on the existing literature [11], a mobile agent must contain all of the following models: an agent model, a lifecycle model, a computational model, a security model, a communication model and finally a navigation model. Also, it must exist in a software environment (called the mobile agent environment) in which it can execute.

Based on these issues and in order to allow developers to model mobile agents, we define a set of coherent and interconnected diagrams. We summarize hereafter the main proposed diagrams.

**The Itinerary Diagram**: several definitions of the concept of an itinerary can be found in the literature, such as: «*itinerary describes the places an agent visits, the tasks/ actions to be performed at each place, and the order of such traversal*», «*an agent's itinerary describes the tasks of the agent and the locations where those tasks are to be performed*» [12].

Based on these definitions and so on, we deduce that the static view of itinerary is composed of: *places*, *tasks*, and *results*. In order to model the mobile agent itinerary we define a new diagram, called *itinerary diagram*. This diagram represents an extension of the UML class diagram by adding new stereotyped classes and graphical notations.

In addition, based on these definitions, we deduce that the itinerary defines the *MA travel schema* and the *mapping of tasks to be performed on locations*. These issues represent the dynamic view of the MA itinerary. In order to model the dynamic aspect of the itinerary, we define two new diagrams: navigation and mobile agent activity diagrams.

**The Navigation Diagram** is responsible for modelling the MA travel schema. This diagram extends the UML statechart diagram by adding new stereotyped action and new graphical notations. The states model the locations while the transitions model the movements of an agent.

**The Mobile Agent Activity Diagram** is responsible for modelling tasks to be performed by MA when it is in running state and the mapping of which places these tasks need to be performed. This diagram extends the UML activity diagram by introducing the concept of location. We propose to attach parameters to each activity (Parametric Activity). These parameters specify the list of locations where this activity can be performed. This set of locations can be updated by adding or removing locations. Figure 1 illustrates the mobile agent activity diagram.

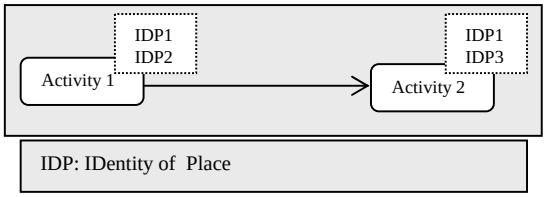


Fig. 1. Mobile Agent Activity diagram

The parameters of an activity make possible to model the reuse aspect of an activity. Also the parameters can be instantiated during the system execution. Then it is possible to model the dynamic aspect of the itinerary. As shown in figure 1, activity1 for example is performed in place P1 and place P2.

**The mobile agent lifecycle diagram**

In order to model the mobile agent lifecycle, we propose a new diagram, called *lifecycle diagram*. The lifecycle diagram extends the UML statechart diagram by adding the concept of location, 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 corresponding to a given location. *The interaction transition* triggers the execution of the *interaction diagram* in a given state in order to execute

communication acts. *The navigation transition* triggers the execution of the *navigation diagram* in a given state in order to determine and to decide to the next location to be visited. Figures 2, 3 and 4 illustrate respectively the graphical notations and the use of activity, interaction and navigation transitions.

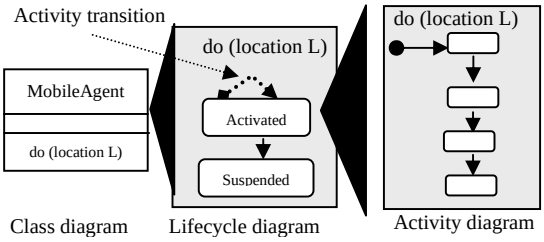


Fig. 2. Granularity levels of the lifecycle diagram use: Activity transition

The *activity transition* labelled with the stereotyped action «do» express that MA change its state having executed a set of activities (tasks) in "activated" state and corresponding to a given location.

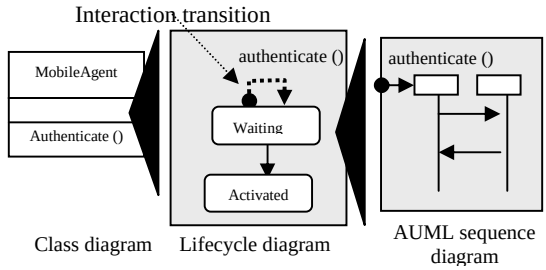


Fig. 3. Granularity levels of the lifecycle diagram use: Interaction transition

The *interaction transition* labelled with the stereotyped action «authenticate» express that MA change its state having executed a set of communicative acts corresponding to the authenticate action.

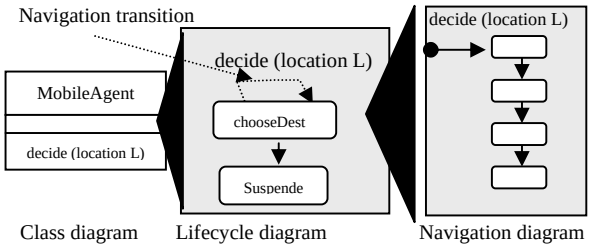


Fig. 4. Granularity levels of the lifecycle diagram use: Navigation transition

The *Navigation transition* labelled with the stereotyped action «decide» express that MA change its state having decide to the next destination to be visited.

## 3.2 Relationship between MA-UML's diagrams

There are seven diagrams in MA-UML notation, which are: environment diagram, mobile agent diagram, itinerary diagram, lifecycle diagram, mobile agent activity diagram, interaction diagram and navigation diagram. Figure 5 illustrates the relationships between the different diagrams, which are defined as follows:

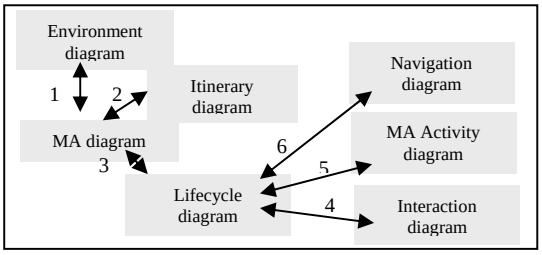


Fig. 5 Inter-Model Relationship

Relationship 1 (Environment diagram ↔ MA diagram): each mobile agent specified in the environment diagram must be specified in the mobile agent diagram in order to specify its internal structural and its characteristics.

Relationship 2 (MA diagram ↔ Itinerary diagram): when identifying the MA internal structure, it is necessary to specify the MA itinerary.

Relationship 3 (MA diagram ↔ Lifecycle diagram): for each mobile agent it is necessary to specify the different states that agent can reach during its lifetime.

Relationship 4 (Lifecycle diagram ↔ Intercation diagram): the lifecycle diagram specify the behavior of MA during its lifetime. During its lifecycle and in a given state, MA needs to communicate and to interact with others entities. This defines the relationship between lifecycle and interaction diagrams.

Relationship 5 (Lifecycle diagram ↔ Activity diagram): in the "activated" state (in the lifecycle diagram) and in a given location, MA must execute the set of tasks related to this location. Therefore, the relationship between the lifecycle and the MA activity diagrams.

Relationship 6 (Lifecycle diagram ↔ Navigation diagram): before to transit in the "migrating" state (in the lifecycle diagram), MA should decide to the next location where it moves. The navigation diagram defines the travel planning. Therefore the relationship between lifecycle diagram and navigation diagram.

## 3.3 Contributions of the MA-UML notation

All approaches previously described in section 2 are useful and practical contributions to model mobile agent applications. However, some deficiencies can be identified; we mention hereafter some of them:

- Mobile agent lifecycle describes all the states that an agent can reach during its lifetime and transitions between states. During its lifetime and in order to achieve its mission, the MA needs to communicate with other entities, 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. In order to specify mobile agent change states, MA-UML proposes new types of transitions in the UML statechart diagram to specify relations with the interaction diagrams, which specifies the communication acts, the navigation diagram, which specifies the travel planning, and the activity diagram, which specifies the tasks plan. Some works have proposed to model the mobile agent lifecycle, but they have not taken into consideration relations between change states and interaction, navigation, and activity models.
- Few works have proposed to design mobile agent itinerary model. However, they lack concepts that specify the relations that exist between locations and activities. MA-UML proposes to introduce the concept of location in the UML activity diagram in order to model relationships between activities and locations by adding parameters to each activity. These parameters represent the set of locations where this activity can be performed. MAM-UML [9] has proposed to model relationships between locations and activities but this work model the static aspect of the travel schema.
- Most works have proposed to model *move* action in UML diagrams, but they did not propose to specify *how* identify the destination address, argument of the move action. In fact, in [13], the syntax of the move action is defined as: <Boolean, reasons> move (location). Where the Boolean value (True or False) indicates whether the movement is successful or unsuccessful, the reason value contains a string message that explains why the movement has failed, and location is the address of location to be visited. MA-UML proposes to model *move* action in the statechart diagram. In order to determine the destination location, MA-UML proposes a new type of transition, called navigation transition, which triggers the execution of the navigation diagram. The returned location represents the argument of the move action.

## 4. Implementation Strategy

In order to support the use of the MA-UML notation, 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 code. In the following sections, we present first the software CASE Tool we have developed to generate Java code automatically from the MA-UML specifications. Then we describe the proposed strategy for mapping mobile agent specifications to Java code.

### 4.1 A CASE tool for mapping MA-UML diagrams into Java code

We developed MAMT (**M**obile **A**gent **M**odeling **T**ool) that is a software development environment to support a mobile agent development process. The MAMT environment was developed as a set of plug-ins for the Eclipse Platform [14]. Eclipse provides as much flexibility as possible and allows developers to create several plug-ins. The Eclipse provides a solid base with the Eclipse Modeling Framework (EMF) and Graphical Editing Framework (GEF).

The MAMT environment, presented in figure 6, consists of three tools:

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

**The translator tool:** includes transformation rules. It is responsible for the transformation of the MA-UML XMI file to the UML XMI file.

**The code generation tool**: includes a Java library. It is responsible to generate automatically Java code based on the UML XMI file.

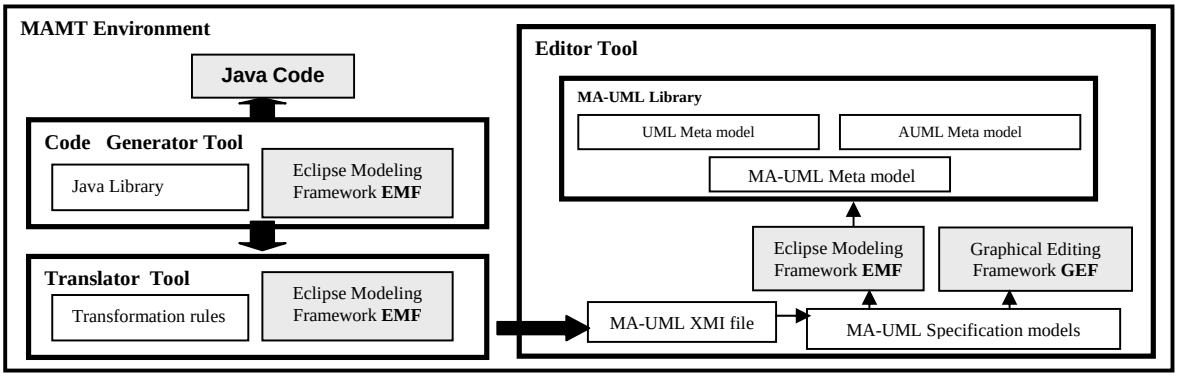


Fig. 6. MAMT environment architecture

### 4.2 Generating Code from MA-UML diagrams

To transform mobile agent specification models into Java code, we propose:

- To transform MA-UML models to MA-UML XMI file. MA-UML XMI is a XMI file generated from MA-UML DTD, which extends the UML DTD according to the extensions proposed by MA-UML to the UML meta-model.
- To transform MA-UML XMI file into UML XMI file by defining a set of transformation rules.
- To generate Java code based on the UML XMI file.

We present hereafter some transformation rules in order to implement the lifecycle diagram.

**R1:** each state in the lifecycle diagram becomes a class.

**R2:** each transition from that state becomes a method in the corresponding class.

**R3:** each action becomes a method in the context class whose behavior is represented by the lifecycle diagram.

**R4:** a context class delegates all events for processing to the current state object.

Figure 7 shows the UML class diagram for implementing MA-UML lifecycle diagram.

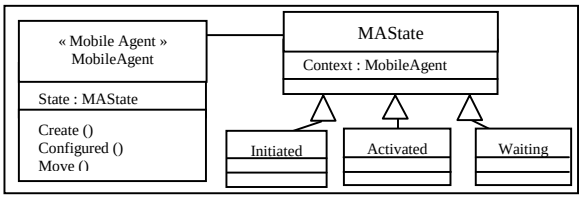


Fig. 7. UML class diagram for implementing lifecycle diagram

## 5. Conclusion

In this paper, an approach for the modelling and the implementation of mobile agents was introduced. It aims to deal with the deficiencies of UML and AUML notations to model mobile agents. This approach is materialized by a UML notation, called MA-UML, and the MAMT software CASE Tool we have developed for mapping MA-UML diagrams into java implementation.

Our future works include two axes. In the first, we are examining how to develop an ontology to act as a specification to allow for the consistency checking of the MA-UML design model. In the second axe we are looking into the design and the implementation of a mobile agent application in the medical field with the MA-UML notation and the MAMT prototype.

## References

[1] OMG. Unified Modeling Language (UML), version 1.4. www.omg.org, September 2001.

[2] Bernhard Bauer, Jörg P. Müller, James Odell: Agent UML: A Formalism for Specifying Multiagent Interaction. In Agent-Oriented Software Engineering, Springer-Verlag, Berlin, pp. 91-103, 2001. (Held at the 22nd International Conference on Software Engineering (ISCE)).

[3] Wooldridge, M., Jennings N. R., and Kinny, D.: The Gaia Methodology for Agent-Oriented Analysis and Design. 2000, Journal of Autonomous Agents and Multi-Agent Systems, Vol.3, No. 3, pp. 285-312.

[4] Arido Y. and Lange D.B.: Agent Design Patterns: Elements of Agent Application Design. *In* Proceeding of Autonomous Agent '98, ACM Press.

[5] G.P. Picco, A.L. Murphy, and G.-C. Roman: LIME: Linda Meets Mobility. In proc. Of 21th Int. Conf. On Software Engineering (ICSE), 1999.

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

[7] Belloni E. and Marcos C.: 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, Volumen 32.

[8] Klein C., Rausch A., sihlinh M. and Wen Z.: 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] Muscutariu F. and Gervais M-P.: On the modeling of mobile agent-based systems. In 3rd International Workshop on mobile agents for telecommunication applications (MATA'01), LNCS Vol. 2164, pp. 219-234. Springer-Verlag, August 2001.

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

[11] Nwana H.: Software agents: An Overview. Knowledge and Engineering Review, 11(3) November 1996.

[12] Ling, S. and Loke, S.W.: Verification of Itineraries for Mobile Agent Enabled Interorganizational Workflow. Proc. of the 4th Intl. Workshop on Mobility in Databases and Distributed Systems. 2001. IEEE Computer Society. (ISBN 0-7695-1230-5). pp. 582-586.

[13] Sutandiyo, W., Chhetri, M, B., Krishnaswamy, S., and Loke, S,W., (2004): Experiences with Software Engineering of Mobile Agent Applications. Australian Software Engineering Conference (ASWEC 2004), Melbourne, Australia, April 13-16, IEEE Press.

[14] Eclipse.org: Eclipse, v 3.0; Available at: http://www.eclipse.org/. Accessed on 05/2005.

**Adlen LOUKIL** received the Dr.degree in Computer Science from Computer Science Department of INSA-Lyon, France in 1996, and was promoted assistant professor in July 199**7.** His research interest includes distributed object, mobile agents, software engineering.