Abstract

The technology of mobile agents obtained recently more importance not only because of its capacity of developing and building a distributed, heterogeneous, and interoperable systems, but also because of its robustness development of mobile and communication network as well. However, there are few works dealing with the methods and tools of analysis and design of the mobile agents systems. Furthermore, the mobile systems have introduced new concepts as: migration, cloning and the locations. We propose in this paper an extension of the most important UML 2.0 diagrams to model the mobile agents systems with the objective to face these three concepts.

Key Words: Mobile Agents, Location, Migration, Cloning, UML, Modelling

1 INTRODUCTION

A mobile agent is a software entity which could move in an autonomous way from a network site to another to reach its objectives. Therefore, the mobile agents induce two important norms of standardization (FIPA [2] and MASIF [3]) supported by a great number of recent platforms [4]. Thus, the mobile agents are introduced in a large number of application fields due to theirs advantages. However, they lack of development methodologies. Obviously it’s advantageous to develop an approach which could be used during all the process of the mobile agent systems development, and inspires its notations from the object-oriented analysis and design systems (in particular UML language). Consequently, many authors proposed some extensions of the UML language to model the mobile agent systems. For example, the proposition of Klein and al [5] is limited to the specific platform GRASSHOPPER. Therefore, this approach can not be generalized to model any application based on mobile agents because of its specific environment. Another extension proposed by Kang et al [6] aims to be an approach uniform with the standards (especially FIPA) to be near to the implementation. It defines for example some modelling elements which referring to the primitives language communication ACL used by the platform JADE [7]. However, the later proposition extends only the Activity diagram of UML 2.0. K.Saleh et al [8] proposed also M-UML which is an extension of UML 1,4 in order to capture the modelling of the new concepts inherent to mobility. It defines, as example, some elements which indicate if a mobile agent is found on another platform different from the basic one, while moving on the different systems sites and interacting with other objects/agents of a system to model as well as the type of their interaction (local or remote). However M-UML is not able to capture implicitly the location of the mobile agent during its movement, which is a basic question in this type of systems modelling. UML 2.0 provides new modelling elements more flexible to capture the new concepts of mobility. For example, the hierarchic multidimensional partitions [where one dimension represents a location and the other represents an object] could be adapted in both state chart and activity diagrams.

The rest of the paper is organized as follows. In section 2 we describe our approach which consists on extending UML diagrams in order to support mobility. In section 3 we present a case study to illustrate our approach. Section 4 concludes the paper and gives some perspectives of this work.

2 MOBILITY AND EXTENDED UML 2.0

Our approach tends to extend the following UML 2.0 diagrams: Use case diagrams, sequence diagrams, class diagrams, collaboration diagrams, object diagrams, state chart diagrams, activity diagrams, and deployment diagrams; with the objective to face the three principal concepts of mobile agents that are location, migration and the cloning. The applicability of the approach will be illustrated through a study case concerning the modelling of the mobile electronic purse.

2-1 The Extended diagrams

In this section and for each diagram we present the new introduced elements.

2-1-1 Use case diagram

To model mobility concepts, we have added to the use case diagram, the notions of mobile actor and migration association.

2-1-1-1 Mobile actor

The notion of mobile actor is shown in figure 1.

Figure 1: mobile actor
A mobile agent could be represented in the Use case diagram as a mobile actor, it is stereotyped by << ma: name of the mobile agent>>. The idea of mobile actor is inspired from M-UML [8].

2.1.1.2 Migration association <<send>>

In this diagram we propose the association <<Send>> as shown in figure 2 and explained below.

- A new association between any actor and a mobile actor: an association stereotyped with <<send>>; shows the original (base) location from which the mobile agent (actor) can move.
- The base location of mobile actor (agent) is the basic platform of the actor from which the relation (arrow) starts.

2.1.2 Sequence diagram

We propose in figure 3 the elements to model the cloning and the migration actions of a mobile agent with regard to the system objects interaction in this type of diagram, as well as the location towards which this agent will be moved.

- <<move to : id-location>> : the mobile agent moves towards the location identified with id-location (the identifier of location could be the name of any object which designs its basic location)
- <<<clone: nbr >>>: the mobile agent clones to an entity number nbr.
- <<<return >> in this case the mobile agent should return towards its original location.

2.1.3 Class diagram

To support mobility, we added to the class diagram the concept of Mobile Agent class (MA).

Mobile Class (MA)

We define the mobile class MA as an abstract class stereotyped with <<mobile>>. This is illustrated by figure 4.

```
<<mobile>>

MA
+ clone(int)
+ moveTo(Location)
+ return()
```

Figure 4: Mobile Class

The Mobile Class MA contains three necessary methods to realize the mobile agent:
- clone(int) with this method, the object mobile agent could clone it self to a determined number entity.
- moveTo(location) : with this method, the object mobile agent could move forward the determined location Figure
- return with this method the object mobile agent could return to its basic platform.

The MA class should always be present when we want to model a system of mobile agents. Then, all the other system mobile classes inherit this class.

2.1.4 Object diagram

The Mobile Object is an instance of the mobile class MA. It is represented in our object diagram by an UML object stereotyped with <<mobile>>:

```
<<mobile >>

ma:MA
```

Figure 5: Mobile Objet

We introduce in the object diagram some elements in order to illustrate the cloning actions, and the movements of a mobile agent in relation to the exchanged messages between the system objects, as well as the locations to which this mobile agent will move:
Figure 6: the cloning action and movement according to exchanged messages

- i: «moveTo: id-location>> : the mobile agent will move to the identified location by (id-location).
- k: «return>> : in this case, the mobile agent must return to its original location.

Where: i, j and k show the order of these actions with regard to other exchanged messages.

2.1.5 Statechart diagram:

UML2.0 provides the multidimensional hierarchical partitions. One dimension represents a location and another orthogonal dimension represents an object. These modelling elements are more flexible to capture new concepts of mobility. We will be able to adapt them easily in both state chart and activity diagrams of our approach.

- In a State chart diagram of our extension, a dimension represents a location which is represented by the following stereotype from: «platform: a name of an agent X (or else an object) » (see figure 7, which designs the basic platform of the agent or the object x).
- The other orthogonal dimensions represent the system objects as well as the mobile agents; to distinguish the mobile agents from the other objects we use the stereotype «mobile ».
- An event provoking the mobile agent to move from any platform to another one (different from the basic one), is labelled with the stereotype «move».

For example (Figure 7) the event Event1 provoke the mobile agent ma to move from id-location1 platform to id-location2 platform to achieve State2. So an event (Event i) provoking a mobile agent to return to its basic location, is labelled with the stereotype «return».

Figure 7: Statechart diagram

2.1.6 Activity diagram

An Activity diagram could contain three types of nodes: action node, object node, and control node as shown:

Figure 7: New elements in Activity Diagram

UML2.0 improved the set of the activities nodes by the “Activity Parameter Nodes”. These objects nodes could be placed at the input/output of an activity and used to describe the flow of an object to the input and the output of an activity. In these limits we propose two modelling elements: the mobile agent node “MobileAgentNode” and the mobile parameter node “MobileParameterNode”.

- A MobileAgentNode is used to model the flow of a mobile agent in activity diagrams; it is represented by an object node stereotyped with «mobile».
- A MobileParameterNode is used to model the flow of a mobile agent at the beginning and the end of activities, it is represented by a parameter node stereotyped also with «mobile». In the following figure (Figure 8): ma1:MA, ma3:MA and ma4:MA represent mobiles parameter nodes, the ma2:MA represents a mobile agent node.

Figure 8: mobile agent and parameter nodes in activity diagram

2.1.7 Deployment diagram

A deployment diagram is useful for modeling a distributed and client/server systems. The diagram shows the distribution of physical processing units or nodes and components residing in them. A node in a mobile system can send and receive mobile agents. In our deployment diagram, a node which is going to send mobile agents is labelled with the stereotype «send» followed by the names of these mobile agents. So a node which is going to receive mobile agents is labelled with the stereotype «receive» followed by the names of these mobile agents.

Figure 7: Statechart diagram
Figure 9: Deployment diagram

- The first node “Node1” can send the mobile agent MA1 and receive the mobile agent MA2.
- The second node “Node2” can receive both mobile agents MA1 and MA2.
- Finally, the node “Node 1” can send MA2 and receive MA1.

3 CASE STUDY: THE MOBILE ELECTRONIC PURSE

To realize a Purse system with mobile agents let’s assume the existence of many shareholders buyers and sellers and each one having a cheque account in a bank level belonging to our system and a share account in a purse level. Whenever, a session Purse is opened the shareholders start handing over their selling and buying orders. The mobile system contains both kinds of agents: stationary and mobile, described as following:

- Three classes of mobile agents: Buyer Mobile Agent BMA, Seller Mobile Agent SMA and Purse Mobile Agent PMA.
- One class of stationary agent: Cheque Accounts Manager Agent CAM, each CAM is situated in a platform of a system bank server, and two other stationary agents: the Share Accounts Manager SAM, and the Transactions Manager TM that are situated in the purse server.

If a buyer (i) places a buying order for a particular share, a mobile agent BMAi will carry this order and move to the platform of the purse, where the TM is situated. The carried orders contain the following information: the shareholder’s buyer identity, the wished share name, the demanded quantity as well as the proposed price. The TM should classify the buying order, if it is accepted, after receiving BMAi. In this way, if a seller (j) places a sale order, it provokes the agent SMAj to move towards the TM platform carrying the selling order with the following information: the shareholder’s seller identity, the exposed share name, the quantity as well as the price needed. At last, the selling order will be classified by the TM as that of the buying orders.

At a determined time, the TM agent calculates the current price of each share offered in the purse session and classifies the best selling and buying orders. Then, the TM sends back the mobile agents BMAs and SMAs coming from the best shareholders orders to consult them to assert or cancel their respective orders. If a transaction is adjusted, the TM agent creates a mobile purse agent PMA that clones itself to a number equal to the number the banks holding cheque accounts of shareholders concerned by this transaction. Then, each PMi (the PM created and PMs cloned) to a well determined platform (on one of these banks) where a CAMi is situated. After coming to the platform of the respective CAMi, PMAi invoke it locally to update viewed cheque accounts; then the CAMi credits accounts of sellers and debits accounts of buyers. In this way, the TM agent invokes the SAM to update share accounts of these shareholders. Finally, after updating cheque accounts, each PMAi should return to its basic location (the platform of the TM agent).

The following figure shows the global architecture of our mobile electronic purse system.

Figure 10: the mobile electronic purse system

3.1 Use case diagram

The following Figure shows the use case diagram of the purse system. This diagram contains three mobile actors BMA, SMA, and PMA.

Figure 11: The use case diagram of the mobile purse
• A mobile actor BMA (see the precedent Figure) is able to be sent from a platform of a buyer in order to participate with the TM actor in the use case Transactions Manager.
• A second mobile actor SMA is able to be sent from a seller platform to participate in the precedent use case.
• A third mobile actor PMA is able to be sent from the platform of the TM agent in to participate in the use case Cheque Accounts Manager.

3.2 Sequence diagram

The following sequence diagram details the use case Buy.

![Sequence Diagram of Buying Order Placing]

The new element stereotyped with <<moveTo: id-location>> allows to capture easily the moving of any mobile agent on its time line. Consequently, we can recognise the locations from which the mobile agent interacts with other objects system. For instance, after moving to the platform of the TM agent (showed in the precedent Figure by <<moveTo: tm>>), the mobile agent bma:BMA can interact locally with the tm:TM agent.

3.3 Class diagram

In this example, we have identified the super class MA and three subclasses (BMA, SMA, and PMA). All these classes are stereotyped with <<mobile>>.

3.4 Collaboration Diagram

![Collaboration Diagram of Placing Buying Order]

The precedent Figure (Figure 14) describes the sequence diagram of the placing buying order (see Figure 12). Then to place a buying order:

• A buyer agent creates a buyer mobile agent bma (1: creat bma());
• Then, the buyer agent invokes it to place the buying order (2: place_order(Order));
• The bma should move to the platform of tm agent (3: <<moveTo: tm>>);
• Finally, bma interacts locally with the tm in order to finish its task (4: classify_order(Order)).

3.5 Statechart diagram

The following Figure shows the statechart diagram of both mobile agents BMA and PMA.

![Statechart Diagram of Mobile Agents BMA and PMA]
The first orthogonal dimension shows somemobile agent states **bma:**BMA connected with transitions. The first state reachable is **ReadyToBuy**; after receiving the message **Buy:**Buy.place_order from the **buy:**Buy object, the **bma** moves to the base platform of the TM agent to reach the state **ClassifyingBuyingOrder**, so it sends the message **TM:**TM.classify.Order to the TM in order to classify its buying order, then the **bma** changes its state to **WaitingForOffer** in which it waits until it receives the message **TM:**TM.offer from the TM; that provokes it to move to its owner shareholder in order to display this offer. An offer consists of the price calculated of the share requested and the available quantity.

### 3.6 Activity diagram

The Figure 16 shows the activity diagram of the share buying. In the “**Place Order**” activity, a buyer tries to place a buying order, then a mobile agent BMA should move by the “**Go**” action to the TM agent basic platform. In the “**Receive Orders**” activity, the TM receives buying and selling orders from BMA and SMA coming from their basic platforms. Two MobileParameterNodes BMA and SMA are placed in inputs of the “**Receive Orders**”. The buying order will then be managed in the “**Manage Orders**”. If the order is accepted [**Order accepted**], the mobile agent BMA should return to its basic platform by the “**return**” action in order to display the current offer. In the “**Make decision**” the buyer can assert [**Ok**] or cancel [**Cancel**] his order. If some buying and share orders are asserted, the “**Update Accounts**” activity will then start in order to update cheque and share accounts of this transaction. The rake resembles a miniature hierarchy indicating that this activity starts another activity that represents a further decomposition.

![Figure 16: The activity diagram of share buying](image)

### 3.7 Deployment diagram

![Figure 17: The deployment diagram of mobile purse system](image)

In this last diagram as the two elements <<send>> and <<receive >> shows:
- The purse server node could send mobile agents of type PMA and receive agents type BMA and SMA;
- A buyer server node could send agents of type BMA;
- And a seller server node could send agents of type SMA.

### 4 CONCLUSION

UML 2.0 offers new modeling elements more flexible, allowing to capture the new mobility concepts. For that, we have proposed in this paper an approach which is an extension of the UML2.0 version, with the aim to cover the three principal concepts of mobility: location, migration and cloning. In order to show the applicability of our approach, we used as a case study: the modelling of electronic purse system.

The behaviour of a mobile agent is defined from its plans and actions. At any moment the mobile agent chooses a set of plans to execute in order to reach its goals. In a future work we attempt to enrich our contribution by introducing new modelling elements to take into account the plans and the actions of mobile agents in one hand and translate mobile UML 2.0 diagrams to their equivalent Petri Nets for analysis purposes using graph transformation on the other hand.

### 5 REFERENCES


http://mole.informatik.unistuttgart.de/mal/mal.html


[10] Dianxiang Xu and Yi Deng, Modeling Mobile Agent Systems with High Level Petri Nets, School of Computer Science Florida International University Miami, FL33199.