Using the UML 2.0 Activity Diagram to Model Agent Plans and Actions

Viviane Torres da Silva[1] Ricardo Choren Noya[2] Carlos José Pereira de Lucena[1]

[1]Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro,
Rua Marques de São Vicente, 225 – Gávea, Rio de Janeiro, RJ, Brasil
{viviane, lucena}@inf.puc-rio.br

[2]Department of Systems Engineering
Military Institute of Engineering (IME)
Pç. General Tibúrcio, 80 – Praia Vermelha, Rio de Janeiro, RJ Brasil
choren@de9.ime.eb.br

PUC-RioInf.MCC48/04 December, 2004

Abstract. The behavior of an agent is defined through the specification of plans and actions. Agents have a set of plans that are selected to be executed according to their goals (and other mental state information). In this paper, we propose the use of UML 2.0 activity diagrams to model agent plans and actions. We consider a plan to be an activity. Both plans and activities are composed of actions and define the action execution sequence. By using some features available in the UML 2.0 activity diagrams and defining some new ones, we demonstrate how these diagrams can be applied to model agent plans and actions.

Keyword. UML, activity diagram, multi-agent system, modeling, implementation

Resumo. O comportamento de um agente é definido através da especificação dos planos e das ações. Agentes possuem um conjunto de planos que são selecionados para serem executados de acordo com seus objetivos (e outras informações em seus estados mentais). Neste artigo, nós propomos o uso do diagrama de atividades de UML 2.0 para modelar os planos e as ações. Nós consideramos um plano uma atividade. Tanto planos como atividades são compostos por ações e definem a seqüência de execução das ações. Utilizando algumas características disponíveis no diagrama de atividades de UML 2.0 e definindo outras, nós demonstramos como este diagrama pode ser aplicado na modelagem de planos e atividades de agentes.

Palavras-chave.UML, diagrama de atividade, sistema multi-agentes, modelagem, implementação

1. Introduction

Agents are software entities designed to satisfy specific conditions called goals. Adopting a goal represents some commitment to pursuing a particular state. While specifying a multi-agent system, designers build plans to determine how the agents rationally act in accordance with their goals.

Currently, there has been an increasing effort to use UML to specify multi-agent systems, e.g. AUML [7], AORML [15] and MAS-ML [13]. Nevertheless, these efforts focus on the structural and interactive aspects of the system. They provide no basis for modeling plans that ensure the achievement of goals and no guidance whatsoever about how plans are related to agent roles, interactions and organizations. These gaps pose significant problems for modeling the dynamic behavior of multi-agent systems.

Usually, agent interaction protocols are used to model the internal features of agents. However, it is also important to specify the high-level business processes, i.e. to model the complex logic, including data flow, within a software agent.

To overcome these limitations, we propose using the UML 2.0 [14] activity diagram to specify action plans. This diagram models the system behavior, including the sequence and conditions of execution of the actions. Actions are considered the basic units of the system behavior. The activity diagram is the most noticeable change in UML 2.0. In fact, this is a completely new diagram. It is not a specialization of a state diagram, but rather a combination of data and object flow diagrams. This way, we intend to provide a notation for plans that indicate the rational achievement of goals, so that the diagrams can show agents playing roles, executing actions and exchanging messages. We also intend to enhance the MAS-ML modeling language.

This paper is structured as follows. We begin by describing some characteristics of multi-agent systems that are significant to justify the use and extension of the features available in the UML 2.0 activity diagram to model plans. Section 3 presents an example of a multi-agent system that will be used to show the modeling features introduced in Section 4. Section 4 presents the features available in the UML 2.0 activity diagrams used to model plans and actions and other new features defined to model plan related characteristics that could not be modeled using the standards features. Section 5 describes the related work, and Section 6 concludes and presents some ongoing work.

2. Agent Basics

Agents are goal-oriented entities that have beliefs, plans and actions defined in its mental state [3][12][10]. Beliefs include what the agent knows about the environment, itself and other agents, and its perceptions about what happens in the system [13].

A plan is composed of actions and defines a way to achieve a goal. A plan can be viewed as state transition machine [5] where the states define the actions that should be executed and the edges link these actions, defining their execution order. The transitions from an action to another can be evaluated according to the information represented in the agent's mental state.

Agents play at least one role in an organization [8][18] and inhabit exactly one environment [2][3]. A role defines duties and rights that an agent must obey while executing [15][16]. Besides defining duties and rights, a role also defines the protocols available to the agents playing it. A protocol describes a sequence of messages that can be sent or received by agents.

While executing, agents can commit to new roles, cancel its roles, deactivate or activate roles, and change from one role to another [9][13]. When agents change their roles, they can move from an organization to another and even from an environment to another.

3. An Example: The Expert Committee MAS

Consider the domain of conference management, where authors can submit papers and a chair distributes these papers among the reviewers for evaluation. The Expert Committee is an application solution developed as an example of multi-agent system for the conference management domain.

In the Expert Committee system, agents play different roles to achieve their goals. The system supports the following activities: paper submission, reviewer assignment, review submission, and acceptance or rejection notification. Throughout this paper, we will focus on the reviewer assignment activity to show examples of the notation we propose.

Until the submission deadline, authors can submit their papers. Once the deadline is reached, the chair must distribute the set of submitted papers to the reviewers according to their research area. The system has a configuration parameter that states that a paper must be reviewed by at least *n* reviewers. The chair keeps trying to allocate the reviewers for a limited period of time, after which, if a paper does not have *n* reviewers, the chair himself becomes responsible for reviewing this particular paper.

The allocation activity is carried out in the following way. The chair sends a paper review proposal to a reviewer. The reviewer then evaluates the proposal to accept or reject it. Each reviewer must tell the chair about the papers he agrees to review.

Agents are used in the Expert Committee application to help the chair with the distribution of papers to reviewers and to assist a reviewer with the evaluation of proposals. The scenarios that will be pointed out in the following section are the *Distribution of papers to reviewers* and the *Evaluation of proposal of reviewing papers*.

4. The UML Activity Diagram

Activity diagrams emphasize the sequence and conditions for action execution. An activity is a specification of parameterized behavior that is expressed as a flow of execution through the sequencing of subordinate units (whose primitive elements are individual actions) [14]. An action represents a single step within an activity, thus being the fundamental unit of behavior specification [14].

The UML 2.0 activity diagram was used to model agent plans and actions. To model agent plans and actions, it was necessary to extend the activity diagram with new stereotypes related to multi-agent system characteristics.

4.1. Plans

The definition of plans and activities are similar. Plans are composed of actions and define the order in which they can be executed, thus activity diagrams can be used to model plans. Like an activity, a plan can be illustrated by using three different representations. In Figure 1 a plan is modeled as a simple activity. The actions and edges that compose and describe the plan are modeled inside a round-cornered rectangle identified by the name of the plan. Figure 1 illustrates the plan *Evaluation of proposal of reviewing papers*.

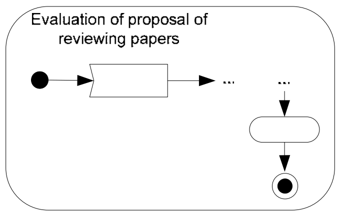


Figure 1. A plan modeled as an activity

Figure 2 illustrates an invoking plan by using the representation of an invoking activity. An invoking plan is a plan that is invoked by another plan or action. The rake-style symbol indicates decomposition or sequence of plans. Figure 3 shows the contents of the invoked plan inside a large round-cornered rectangle by representing the edges and nodes of the plan.

Figure 2 and Figure 3 depict the plan *Distribution of papers to reviewers*. This plan was modeled as an invoking plan since it is called up by the action that monitors the submission deadline. Once the deadline is reached, the chair executes the plan *Distribution of papers to reviewers*.

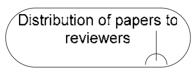


Figure 2. Example of invoking a plan

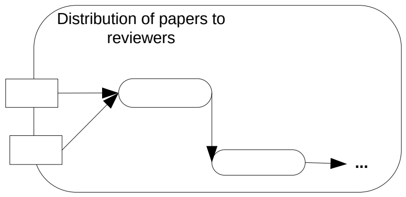


Figure 3. The nodes and edges of a plan

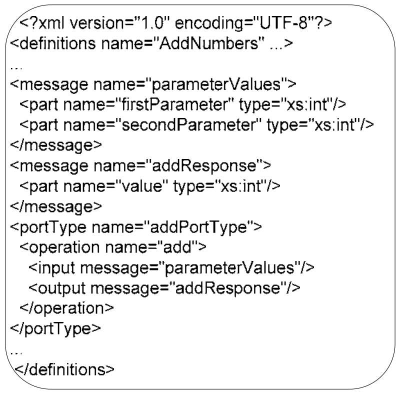
4.2. Actions

In UML activity diagrams, it is possible to define an action in two different ways. An action can be identified only by its name or it can be described using an application description language. We propose to describe an action by using a domain-independent notation.

We believe that actions can be viewed as components. When the designer defines an action, he is specifying a component that will implement a given functionality. Besides, a plan is just a logical sequence of actions. The implementation of these actions can be independent, to maximize loose coupling and action reuse.

To illustrate this idea, actions could be described and implemented using a services approach. In such approach, all actions are defined as services, which are seen as black boxes , i.e. external actions or plans neither know nor care how they perform their functionality. In a more general sense, the action interface is invokable. This means that it is irrelevant if an action is local (within the system) or remote (external to the immediate system), what interconnect scheme or protocol is used to effect the invocation, or what infrastructure components are required to make the connection.

In this approach, an action could be further specified using Web Services Description Language (WSDL) [6]. Nonetheless, the WSDL description of any action can become really extensive. Figure 4 illustrates parts of the description of a simple action that receives two integers and returns the sum of such integers. It is possible to notice how vast the description of actions can become by using WSDL. To solve such problem, we propose to identify actions by describing their names and identifying the URLs where their WSDL descriptions are available. Figure 5 illustrates the action *AddNumbers*.



```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="AddNumbers" ...>
...
<message name="parameterValues">
  <part name="firstParameter" type="xs:int"/>
  <part name="secondParameter" type="xs:int"/>
</message>
<message name="addResponse">
  <part name="value" type="xs:int"/>
</message>
<portType name="addPortType">
  <operation name="add">
    <input message="parameterValues"/>
    <output message="addResponse"/>
  </operation>
</portType>
...
</definitions>
```

Figure 4. Action described using WSDL

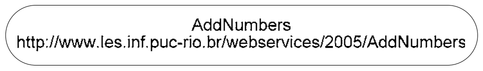


Figure 5. Target namespace identifying the action

4.3. Goals

Agents execute plans in order to achieve theirs goals. The plans of an agent are associated with its goals; therefore, while modeling plans, the goals associated with them should be identified. We propose the use of a new stereotype <<`goal`>> to describe the goal related to a plan. Figure 6 shows the use of the stereotype <<`goal`>> to associate the goal *Evaluate proposal* with the plan *Evaluation of proposal of reviewing papers*. The goal *Evaluate proposal* is one of the goals of the reviewers. In order to achieve such goal, reviewers execute the plan *Evaluation of proposal of reviewing papers*.

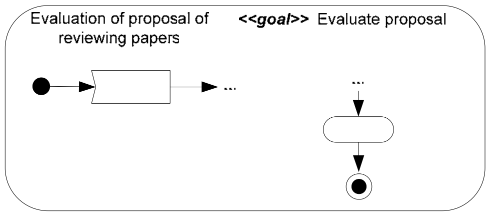

Evaluation of proposal of reviewing papers
<<goal>> Evaluate proposal
...
...

Figure 6. Relating goals to plans

4.4. Guard Conditions in Decision Nodes

Guard conditions are defined in the decision nodes of activity diagrams to describe conditions that must be satisfied in order to fire an associated transition. We propose to extend the definition of the guard conditions to describe information related to the agent's mental states. Such information may describe the conditions that should be checked by the agent in order to decide the next action to execute.

Agents can decide whether or not to execute an action based on any information contained in their mental state. For instance, an agent can decide to execute an action based on the messages that it has received or sent, on the actions that it has previously executed, and on the goals it has. The history about what the agent has done is stored in the agent beliefs.

Figure 7 illustrates the use of beliefs in guard conditions. When the reviewer receives a set of proposal papers to review, the reviewer checks the deadline of the revision according to the dates stored in its agenda. The agent's agenda is one of its beliefs. If the agent realizes that it will be impossible or very difficult to review any paper until the deadline, the agent rejects the proposal.

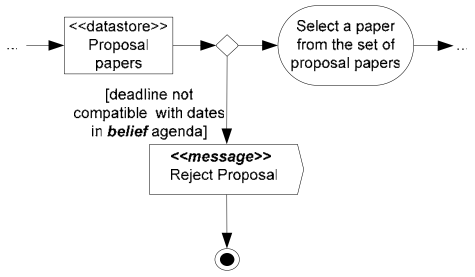

...
<<datastore>> Proposal papers
Select a paper from the set of proposal papers
...
[deadline not compatible with dates in belief agenda]
<<message>> Reject Proposal

Figure 7. Guard conditions

4.5. Message

The UML meta-model defines the *SendSignalAction* and the *AcceptEventAction* meta-classes to represent signals sent to an entity and events received by an entity in activity diagrams. We propose to use such meta-classes to represent the messages sent and received by an agent. To identify the signals and events that are agent messages, the stereotype `<<message>>` should be used. Since messages are sent and received in the context of protocols, it is also important to describe the protocols while identifying the messages.

FIPA ACL parameters can be used to detail the message definition. We do not encourage the designer to identify all the parameters that describe a message while modeling it in an activity diagram. However, we do encourage the designer to select some parameters to help the diagram users to understand some specific and/or important characteristics of the message. Figure 8 illustrates the three proposed representation of messages: (i) simple identification of messages, (ii) identification of message protocols, and (iii) briefly description of messages. The message described in Figure 8 in the one sent by the reviewers to the chair when they reject the proposals of reviewing papers.

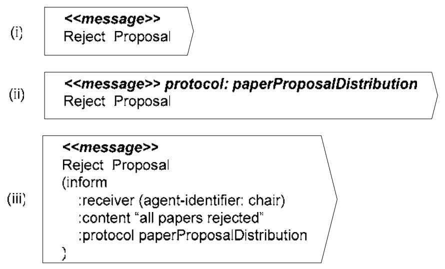

(i)
<<message>>
Reject Proposal
(ii)
<<message>> protocol: paperProposalDistribution
Reject Proposal
(iii)
<<message>>
Reject Proposal
(inform
:receiver (agent-identifier: chair)
:content "all papers rejected"
:protocol paperProposalDistribution
)

Figure 8. Representing messages

4.6. Roles

As stated before, an agent is always playing at least one role. To specify the roles an agent is currently playing, they should be identified during the plan modeling. We introduce of the stereotype `<<role>>` to define the possible roles an agent can be playing while executing a plan. Figure 9 shows the use of the stereotype to state that the plan *Evaluation of proposal of reviewing papers* is executed in the context of the role *reviewer*.

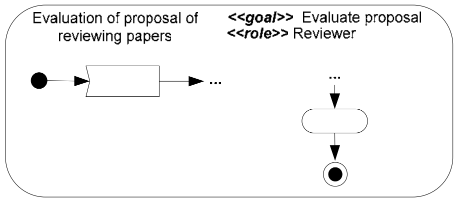

Evaluation of proposal of reviewing papers
<<goal>> Evaluate proposal
<<role>> Reviewer
...
...

Figure 9. Relating roles to plans

While executing a plan, agents can play different roles. An agent can commit to a new role, can cancel one of its roles, can temporarily stop playing a role (deactivating it) or can activate a role that it has temporarily stopped playing. Therefore, besides identifying roles while executing plans, it may be useful to relate the roles to the actions of a plan.

To represent agents changing roles, we suggest the use of stereotypes associated with the actions where these changes take place. The stereotype `<<role_commitment>>` should be used to model an agent committing to a new role, the stereotype `<<role_cancel>>` to model an agent canceling a role, the stereotype `<<role_deactivate>>` to model an agent temporarily stooping playing a role and the stereotype `<<role_activate>>` to model an agent activating a role.

Figure 10 illustrates the use of swimlanes and the stereotypes `<<role_commitment>>` to model two different roles played by an agent. While executing the plan *Distribution of papers to reviewers*, the agent playing the role *chair* may need to commit to the role *reviewer* if a paper has not been associated with reviewers until the papers distribution deadline. The agent commits to the role *reviewer* while executing the action *Allocate papers without reviewers to agent*. Note that the proposed notation, with stereotypes and swimlanes, handles concurrency aspects of role playing. In this particular example, the agent does not stop playing the role chair: it starts playing the role receiver simultaneously.

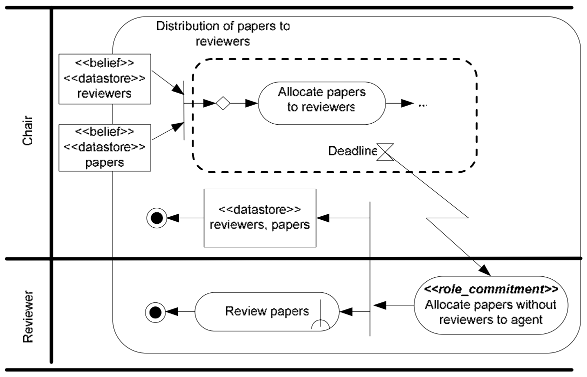


Figure 10. Partitions, roles and actions

In order to demonstrate the interruption of an action execution inside the *Distribution of papers to reviewers* plan, the diagram element *Interruptible Activity Region* proposed in the UML 2.0 meta-model was used. Figure 10 illustrates that the agent playing the role *chair* interrupts the execution of its plan to commit to the role *reviewer*. Besides, Figure 10 depicts the use of the stereotype `<<belief>>` to indicate that the list of reviewers and papers are beliefs of the agent playing the role chair.

We also introduce the use of partitions. Partitions often correspond to organizational units in a business model [14]. This way, actions can be either separated into groups or annotated to identify their group. Figure 11 shows the use of partitions by annotating the name of the role that is being played when an action is executing.

The use of annotations and stereotypes are interesting to illustrate an agent changing roles. When an agent wants to change its role, it can cancel or deactivate its previous role and commit to or activate a new role. Figure 11 demonstrate how to model these four situations by using stereotypes and partitions.

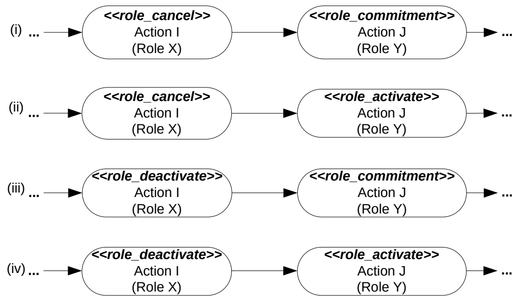

(i) ...
<<role_cancel>> Action I (Role X)
<<role_commitment>> Action J (Role Y)
...
(ii) ...
<<role_cancel>> Action I (Role X)
<<role_activate>> Action J (Role Y)
...
(iii) ...
<<role_deactivate>> Action I (Role X)
<<role_commitment>> Action J (Role Y)
...
(iv) ...
<<role_deactivate>> Action I (Role X)
<<role_activate>> Action J (Role Y)
...

Figure 11. Changing roles

4.7. Duties and Rights

In the previous section, we have used partitions to relate actions and the roles played by agents while executing those actions. Since roles and actions are related, it is also possible to identify actions that are the duties and the rights of the agent playing the role.

We introduce the stereotypes `<<duty>>` and `<<right>>` to describe the duties and rights of an agent according to the role, respectively. Figure 12 illustrates the use of the stereotype `<<duty>> and <<right>>`. The action *Allocate papers to reviewers* is one of the duties of the chair role while distributing papers to reviewers and the action *Reject paper to review* is one of the rights of the reviewer role while evaluating the proposals of reviewing papers.

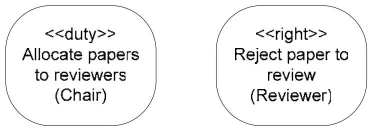
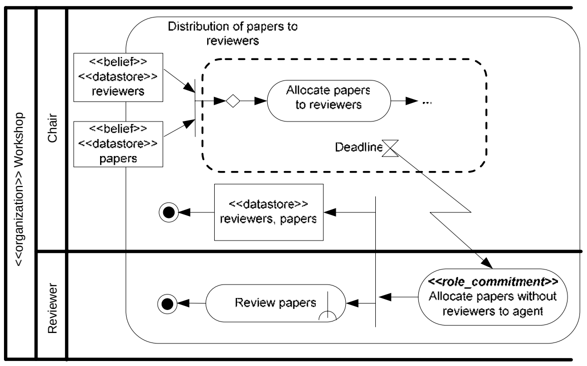

<<duty>> Allocate papers to reviewers (Chair)
<<right>> Reject paper to review (Reviewer)

Figure 12. A duty action

4.8. Organizations

Agents play roles in the scope of an organization. Moreover, an agent can change from an organization to another to play a different role. Thus it is important to identify the organizations to model such features of the system. We propose the use of partitions and the stereotype `<<organization>>` to identify organizations.

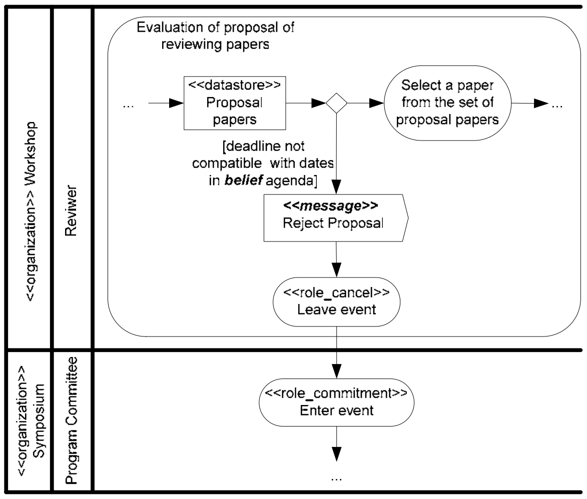
Using partitions and such stereotype it is possible to model agents changing their roles in the same organization and agents changing their roles when moving from an organization to another. Figure 13 illustrates an agent committing to another role in the same organization and Figure 14 depicts an agent moving from an organization to another by canceling its previous role and committing to a new role in another organization. In Figure 13 the agent playing the role chair commits to play the role reviewer in the same organization where it is playing the role chair.



<<organization>> Workshop
Chair
Reviewer
Distribution of papers to reviewers
<<belief>> <<datastore>> reviewers
<<belief>> <<datastore>> papers
Allocate papers to reviewers
Deadline
<<datastore>> reviewers, papers
<<role_commitment>> Allocate papers without reviewers to agent
Review papers

Figure 13. Playing two roles in the same organization

In the Expert Committee system, if the agent playing the role *reviewer* does not agree to review a paper because of the revision deadline, it can stop playing the role *reviewer* in that event. For example, Figure 14 illustrates the agent canceling the role *reviewer* in a *Workshop* and committing to the role *program committee* in a *Symposium*. The agent leaves the workshop organization and enters into the symposium organization.



<<organization>> Workshop
Reviwer
<<organization>> Symposium
Program Committee
Evaluation of proposal of reviewing papers
<<datastore>> Proposal papers
Select a paper from the set of proposal papers
[deadline not compatible with dates in belief agenda]
<<message>> Reject Proposal
<<role_cancel>> Leave event
<<role_commitment>> Enter event

Figure 14. Changing organizations

4.9. Environments

Mobile agents can move from an environment to another while executing their actions. In order to represent agents moving from an environment to another, the environments where agents are executing should be represented in activity diagrams. We introduce the stereotype `<<environment>>` to identify environments and the use of partitions to model agents changing environments.

To move from an environment to another, the agent stops playing roles in organizations of the departure environment and starts playing roles in organizations of the arrival environment. The hierarchical partition notation can be used to model environments, organizations and roles. Hierarchical partitions represent the children in the hierarchy as further partitions of the parent partition [14]. Roles' partitions are children of organizations' partitions since different roles are played in organization. Besides, organizations' partitions are children of environments' partitions since organizations inhabit environments.

Figure 15 shows an agent moving from the environment *Env1* to the environment *Env2* by deactivating the role *Z* in organization *A* and committing to the role *W* in organization *B.*

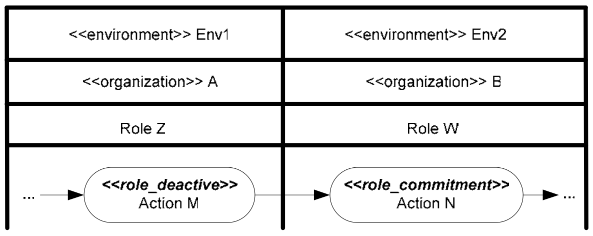


Figure 15. Changing environments

5. Related Work

The UML activity diagram has been used to model the behavior of mobile agents [1][4], to model agent interaction protocols [6][11] and to model agent plans [5]. By using stereotypes, they extend the activity diagram to model, for instance, roles, environments, messages and organizations.

Kinny and Georgeff in [5] use the UML activity diagram to model agent plans. They define internal states as activities that are related to sub-goals and they propose to associate conditions with the agent beliefs. In addition, they define fail states that are introduced to model the notion of failure. However, they do not use activity diagrams to model the relationship between plans, goals and roles played by the agent. Therefore, it is not possible model all the organizational aspects by using their approach. Moreover, they also do not model agent messages.

In [4], the authors use the UML 2.0 Activity Diagram just to model some specific behaviors of mobile agents, which are cloning, mobility and message passing. The mobility feature is modeled merely as a "Go - Do Task" activity pair, each one executed in a host. We are more concerned with the organizational aspects of mobility, providing a way to model agents changing roles, stopping action plans and moving from one organization to another, which may not even be in different hosts.

In their approach, message passing is represented with signal sending and signal receipt, combined with ACL performative stereotypes. We also use signals and ACL to model messages but we go further in specifying the messages by identifying the protocols, the receivers and any other relevant message information that can be described by using ACL. The work presented here does not provide special notation for cloning since it can be simply modeled as an activity (or a set of activities) in an agent action plan.

In [1], the authors propose the use of stereotypes associated with actions to model mobile objects, locations, mobile locations and actions that move and clone agents. In this paper, we do not propose any specific stereotype to be used associated with mobile agents. The only stereotype that is related to mobility characteristic is the stereotype `<<environment>>` used together with partitions to model environments.

Lind [6] recommends the use of the stereotypes `<<send>>` and `<<receive>>` to model messages. Those stereotypes are identified in the edges that link actions to indicate the actions that send or receive messages. Since our proposal extends the UML 2.0, in this paper we suggest to indicate messages by associating stereotypes with the *AcceptEventAction* and *SendSignalAction* meta-classes. Such meta-classes were not available in previous UML versions.

In [6] the authors also propose to model roles by using the stereotype `<<roles>>` related to swimlanes. Nevertheless, in our approach, we also model the modification of the agent roles by associating stereotypes related to the actions that originate the change.

In [11] the authors use swimlanes to model roles and group of agents (or organizations). They also suggest the use of the stereotype `<<role change>>` in notes related to actions to indicate when an agent changes its roles. We extend their proposal defining other stereotypes to point out how the changes occur. We also suggest the use of partitions to model agents moving from an organization to another.

6. Conclusion and Ongoing Work

In this paper we propose the use of the UML 2.0 activity diagrams to model agent plans and actions. By using our approach it is possible (i) to describe actions using a domain-independent notation, (ii) to associate goals and roles with plans, (iii) to check the information in the agent mental state by using guard conditions, (iv) to describe messages, (v) to represent agents changing their roles, (vi) to describe the actions that are duties and rights, (vii) to model agents moving from an organization to another and (viii) to model agents moving from an environment to another. The extended activity diagram was included in the set of diagrams proposed by MAS-ML. Nowadays, MAS-ML has tree structural diagrams (extended class diagram, organization diagram and role diagram) and two dynamic diagrams (extended sequence and activity diagrams) that can be used to model the static and dynamic characteristics of agents, organizations, roles, environments and objects.

We are in the way of analyzing the UML 2.0 activity diagram to model the selection of plans. Before executing a plan, the agent must select the plan to be executed from a plan library according to, among other things, the goals it wants to achieve. We believe that the selection of plans can also be modeled as a state transition machine and, therefore, can be modeled in activity diagrams.

Moreover, the interaction overview diagram is also being considered to model plans and their actions. The main difference between this diagram and the activity diagram is that it promotes overview of the control flow. By using this diagram to model a plan, the actions of the plan can be detailed by using interaction, i.e., by identifying the interaction (sequence) diagrams that represent those actions.

References

[1] H. Baumeister, N. Koch, P. Kosiuczenko, and M. Wirsing. “Extending Activity Diagrams to Model Mobile Systems.” In: M. Aksit, M. Mezini, and R. Unland, editors, Objects, Components, Architectures, Services, and Applications for a Networked World, LNCS 2591, pp. 278-293, 2003.

[2] M. d’Inverno; M. Luck. Understanding Agent Systems. New York: Springer. 2001.

[3] I. Ishida, L. Gasser, M. Yokoo. “Organization self design of production systems.” In: IEEE Transaction on Knowledge and Data Engineering, v.4, n.2, p.123-134. 1992.

[4] M. Kang, K. Taguchi. “Modelling Mobile Agent Applications by Extended UML Activity Diagram” ICEIS (4), 519-522, 2004.

[5] D. Kinny and M. Georgeff, (1997) "Modeling and design of multi-agent systems," In Intelligent Agents III, Müller, J., Wooldridge, M. and Jennings, N., Eds., Springer 1193, pp. 1-20.

[6] J. Lind. Specifying Agent Interaction Protocols with Standard UML, In Proceedings of the Second International Workshop on Agent-Oriented Software Engineering (AOSE-2001), LNCS 2222, Springer-Verlag, 2002

[7] J. Odell, H. Parunak and B. Bauer. “Extending UML for Agents” In: Wagner, G., Lesperance, Y. and Yu, E. Proceedings of the Agent-Oriented Information Systems Workshop, AOIS 2000, Eds., Austin, pp. 3-17, 2000.

[8] J. Odell, H. Parunak, M. Fleisher. “The Role of Roles in Designing Effective Agent Organizations.” In: A. Garcia, C. Lucena, F. Zamboneli, A. Omicini, J. Castro, (Eds.) Software Engineering for Large-Scale Multi-Agent Systems. LNCS 2603, Berlin: Springer, 2003.

[9] J. Odell, Parunal, H., S. Breuckner, M. Fleischer. “Temporal Aspects of Dynamic Role Assignment,". In: Agent-Oriented Software Engineering (AOSE) IV, P. Giorgini, Jörg Müller, James Odell, eds., Lecture Notes on Computer Science volume (forthcoming), Springer, Berlin, 2004.

[10] OMG. Agent Platform Special Interest Group: Agent Technology. In: Green Paper. Version 1.0, 2000.

[11] H. Parunak and J. Odell. “Representing social structures in UML” In Agent-Oriented Software Engineering II, Wooldridge, M., Weiss, G. and Ciancarini, P., Eds., LNCS 2222, Springer-Verlag, Berlin, pp. 1-16, 2002.

[12] Y. Shoham. “Agent-Oriented Programming.” Artificial Intelligence, v.60, 1993.

[13] V. Silva, C. Lucena. “From a Conceptual Framework for Agents and Objects to a Multi-Agent System Modeling Language”, In: Sycara, K., Wooldridge, M. (Edts.), Journal of Autonomous Agents and Multi-Agent Systems, Kluwer Academic Publishers, ISSN 1387-2532, pp. 145-189, volume 9, issue 1-2, 2004.

[14] UML: Unified Modeling Language Specification, version 2.0, OMG. Available at: <http://www.uml.org>. Accessed in: December 10, 2004.

[15] G. Wagner, “The Agent-Object-Relationship Metamodel: Towards a Unified View of State and Behavior”, Information Systems, 28(5), 2003.

[16] L. Yu, B. Schmid. “A Conceptual Framework for Agent-Oriented and Role-Based Work on Modeling.” In: G. Wagner, E. Yu (Eds.). Proceedings of the 1st International Workshop on Agent-Oriented Information Systems, 1999.

[17] WSDL. Web Services Description Language (WSDL), version 1.1, W3C. Available at: <http://www.w3.org/TR/wsdl>. Accessed in: December 10, 2004.

[18] F. Zambonelli, N. Jennings, M. Wooldridge. “Organizational abstractions for the analysis and design of multi-agent systems.” In: P. Ciancarini, M. Wooldridge (Eds.) Agent-Oriented Software Engineering, LNCS 1957, Berlin: Springer, p. 127-141. 2001.