Applying and Validating a UML Metamodel for the Requirements Analysis in Multi-Agent Systems: 
The AME-A Case Study

Gilleanes Thorwald Araujo Guedes, Rosa Maria Vicari
Instituto de Informática - Programa de Pós-Graduação em Computação (PPGC)
Universidade Federal do Rio Grande do Sul (UFRGS) - Porto Alegre - RS – Brasil
gttaguedes@inf.ufrgs.br, rosav@inf.ufrgs.br

Abstract—This paper presents the application and validation of a UML metamodel developed for the requirements modeling in multi-agent systems projects. In this paper we describe the metamodel and its application into the modeling of an intelligent tutor system called AME-A. We also demonstrate the way we intend to validate this metamodel.

Keywords-Metamodels; Stereotypes; Agents; Agent Roles; AgentRole_Actors; Internal Use-Cases; AME-A

I. INTRODUCTION

AOSE (Agent-Oriented Software Engineering) is a new area that mixes concepts both from the software engineering and artificial intelligence areas. This new area arose from the multi-agents systems (MAS) development increase and from the new challenges relative to the analysis and project generated by this kind of software. The AOSE seeks to develop methodologies, techniques, and languages that allow the modeling of the special features presented by the MAS.

UML (Unified Modeling Language) is a standard modeling language adopted by the software engineering area, thus, within the AOSE context, some attempts to create new languages after the UML as adapted to the Multi-agent Systems (MAS) project were made. However, none of the languages studied was concerned about extending or adding new metaclasses to the metamodel that serves as the basis for the use-cases diagram (UCD), for that is mainly employed in the requirements analysis (RA) an essential step for the achievement of a good software project.

In this paper we shall discuss the UML-derived languages for the MAS project we have studied. Next we shall speak about the RA importance, passing on to the definition of metamodels, models, and profiles. After that, we shall describe the developed metamodel and present an application of it. Finally, we shall discuss the validation of the metamodel, when we shall compare the diagrams produced using our metamodel concepts with the diagrams produced by means of the standard UML, thus trying to demonstrate that the UML does not support the modeling of the concepts that our metamodel does.

II. DERIVED LANGUAGES FROM UML FOR THE MULTI-AGENT SYSTEMS PROJECT

Some languages were derived from UML to adapt it to the MAS Project, like AUML [1], AORML [2], AML [3], MAS-ML [4], and the [5] agency metamodel. However, there were found no mechanisms in these languages to model the requirements for the MAS project, nor any concern into making a RA as in the system life cycles described by [6] or in software development methodologies like the UP [7].

One of AUML [1] main contributions was the adaptation of the UML sequence diagram to the agent communication. However, there has been no adaptation of the UML for it to be applied into the RA. We imagined that, as the AUML is based in UML, the UCD would be applied in its original way; however, were this the case, this diagram could not be used to model the internal requirements of a MAS.

The AORML concentrates on the matter of interaction modeling between agents and how these react to stimuli. With respect to the requirements modeling, [2] proposes, but does not show, the use of the activity diagram for the RA. In [8] the UCD is applied in its original format, identifying only external requirements; no adaptation of this diagram was attempted to model agent roles, goals, perceptions, actions or plans.

The AML represents agents and the roles played by them and it allows to dynamically model role changes. AML further represents the agents’ perceptions and actions. [3] mentions it is possible to apply mental states so as to enrich the use case modeling by means of the requirements modeling based on goals; however we did not find examples of how that could be done. Besides, no metaclasses were created to model requirements in MAS as proposed in our work.

MAS-ML [4] represents agents, their environments, the resources the agents can manipulate, the organizations of the system and the roles that can be played by the agents, as well as their beliefs, goals, and plans. However, the language is not concerned about the matter of the requirements analysis and no attempt was made to adapt the UCD to this end.

The metamodel in [5] represents roles, organizations, environments, beliefs, goals, and plans, as well as the agents communication. However, this metamodel does not focus on the requirements modeling and no attempt was made to adapt the UCD for this function.
III. REQUIREMENTS ANALYSIS

According to [9], about half of the factors associated with successful projects is related to requirements and the project success is directly bound to its quality. Now [10] states that the incorrect requirements definition is one of the main factors contributing for the failure of a project. And, according to [11], the requirements elicitation, analysis and documentation in complex systems is a crucial and non-trivial task. Finally, according to [6], use-cases (UC) are of special interest in the requirements engineering, since they were demonstrated as valuable to elicit, document, and analyze requirements.

Within this context, several authors recommend the UML UCD for the requirements modeling, like [6], [9] and [11]. The objective of this diagram, as can be seen in [12] is that of identifying the actors that might interact with the system, the functionalities offered by the software and which actors will be allowed to use which functionalities. Being a standard language widely accepted and used, with great flexibility for adaptation to new dominions we believe that UML notation use can eased understanding by developers. Thus we developed a UML metamodel in which we created new metaclasses and stereotypes with the goal of allowing the employment of the UCD for the requirements modeling in the MAS project.

IV. UML, METACLASSES, MODELS AND STEREOTYPES

UML is a visual language for specifying, constructing, and documenting system artifacts. It is a general-purpose modeling language that can be applied to all application domains [13]. The UML specification is defined by using a metamodeling approach. When metamodeling, a distinction is established between metamodels and models. A metamodel defines a semantics for the way of modeling elements within a model. A model captures a vision of a physical system, it is an abstraction of the software with a purpose that determines what must be included and what is irrelevant.

A stereotype is a limited type of metaclass that can only be used in conjunction with one of the metaclasses it is extending. A stereotype allows to define how a metaclass can be extended, assigning to it new characteristics and/or constraints. When a new metaclass is derived from a previous metaclass and uses the stereotype named, “stereotype”, this new metaclass allows to model instances of the original metaclass containing a stereotype with the new metaclass name, making possible to model stereotyped elements in a model with characteristics that differentiate them from the original elements.

V. THE PROPOSED UML METAMODEL

Considering that UML is a widely accepted language and that MAS has its own characteristics and that none of the studied languages that extended the UML to the MAS project focused on the RA issue, we created a metamodel containing new metaclasses and stereotypes prepared to identify the requirements of this kind of system.

Initially, we tried to create a UML profile to adapt the UCD, but we realized that only extending the metaclasses used by this diagram would not be enough. Previous versions of the original profile have been published in [14], [15] and [16]; nonetheless, we evolved it into a real metamodel, adding new metaclasses and stereotypes and adapting it to the work by [17]; besides, in the previous publications we did not speak about how we intend to validate this metamodel.

Initially, we tried to use the Actor metaclass to model agent roles, because, according to [12], this metaclass represents a kind of role played by an entity that interacts with the system, but is external to it. However, the agents are not external to the system, they are inserted within the system and, as they are independent, proactive, and able to interact with the software according to their goals, the roles of these agents must be represented as actors. Therefore, as states [18], it is necessary to adapt this concept, since agents are internal to the system, and if we intend to represent agent roles as actors, these actors must be represented internally, inside the system boundaries.

The representation of agents/roles as UML actors can also be seen in [19], where agents represent active objects and are modeled within the system as actors with square heads. However, [18] only suggests that the actor concept should be adapted, all the while [19] did not create a UML metamodel. In our work we implicitly created new metaclasses to allow for an adequate requirements modeling to be employed in the MAS.

All the same, according to [13], it is not possible to take away any constraints once applied to a metamodel in a profile. Thusly, instead of extending the Actor metaclass, we created a new metaclass from the same metaclass as was created the metaclass Actor, as can be seen in the Fig. 1, thus creating the metaclass AgentRole_Actor. In this new metaclass we copied all the characteristics of the metaclass Actor, suppressing only the constraint that an actor must be external to the system.

From this new metaclass we derived the Reactive and Cognitive_AgentRole metaclasses. We applied the stereotype named, “stereotype”, to both metaclasses, which means that these metaclasses will be applied as stereotypes upon AgentRole actors and will assign them special characteristics.

We specialized still further the Cognitive_AgentRole metaclass by deriving the PS, UAM and SMI_AgentRole metaclasses. They were created after the work of [17], who suggested the use of specialist agents Problem Solving (PS), that must own knowledge about the problem the application will help to solve; Users and Agents Modeling (UAM), that must own the knowledge about the users and agents modeling in order to make cognitive models; and Social Mediated Interactions (SMI), that must own knowledge about how to mediate social interactions between the user and the system.

According to [12], UC can be used to specify the system's external requirements. A UC is the specification of a set of actions performed by software, generating an observable result and producing some value for one or more actors. We intended to represent actions, perceptions, goals, and plans as UC, but the UseCase metaclass semantic states that UC represents external requirements, thus, with the objective of adapting the UC concept for the MAS modeling, we derived a new metaclass from the same metaclass as the UseCase metaclass had been derived, calling it InternalUseCase (IUC) and then we created relationships for this metaclass, equal to those existing between the Classifier metaclass and the UseCase metaclass.
All IUCs are internal to the system and cannot be seen or used by external entities. From the InternalUseCase metaclass, we extended some metaclasses to attribute special characteristics to the IUCs. These extended metaclasses will be employed as stereotypes. Thus, we created the Perception and Action metaclasses to model IUCs that contain the necessary steps for an agent to perceive an event or perform an action.

A third metaclass was derived from the InternalUseCase metaclass, to represent goals. An IUC employing the Goal stereotype will contain a description of an agent’s desire and the possible conditions for that desire to become an intention. A somewhat similar proposal to represent goals as use cases can be seen in [19]. However, besides creating the IUC concept, we went even further, when we considered that, in the same way a goal represents a desire that will not necessarily become an intention, the steps for its execution should be detailed in one or more IUCs other than the IUC that represents the goal.

So, in a situation where an IUC employing the Goal stereotype was used, we would detail in this IUC only those perceptions and conditions necessary for that goal to become an intention. A somewhat similar proposal to represent goals as use cases can be seen in [19]. However, besides creating the IUC concept, we went even further, when we considered that, in the same way a goal represents a desire that will not necessarily become an intention, the steps for its execution should be detailed in one or more IUCs other than the IUC that represents the goal.

According to [12], an extend is a relationship that specifies how and when a behavior defined in the extended use case can be inserted into the behavior of the extending use case. If the extension condition is true, the extended use case behavior will be also performed. Considering that a plan can eventually own more than a plan and that these plans only will be executed under certain conditions, we decided to derive a fourth metaclass named Plan, to identify the plans associated to the goals.

VI. A CASE STUDY - THE AME-A PROJECT

The AME-A architecture [20] is composed by a hybrid society of agents that cooperate into aiding students’ learning. The environment interacts with human agents that can be both the teacher or the students and has several reactive and cognitive agent roles. Previous versions of this case study were published in [15] and [16]; however, in these publications we used old versions of the current metamodel that did not support the types of agent roles proposed by [17].

The teacher can create a learning activity or evaluate the students with the aid of the agent who assumes the Teacher's Tools agent role. The student can choose between performing an unmonitored or a monitored learning session. In the first, he only interacts with the agent who takes the Unsupervised Learning agent role that only presents the content to be learned. The monitored learning activity is the system main focus, in which it aims to maximize the student learning by means of the aid of five cognitive agent roles, to wit: Student Modeling (SM), Methodology and Teaching Plan (MTP), Learning Orientation (LO), Learning Analysis (LA) and Knowledge Application Orienting (KAO). The first models the student profile in a dynamic way, while the second chooses the methodology and teaching plan that are more adequate to the
student profile every time it changes or whenever the student performance is lower than the expected level; the LO agent role selects the contents to be taught and the way how these will be presented according to the methodology; the LA agent role checks on the student performance throughout the session and the KAO agent role applies an evaluation after the session ends.

Considering that teacher and student are both roles assumed by external human agents, we modeled them as normal actors outside the system boundaries and we associated to that actor who represents the teacher the functionalities “Create learning activity” and “Evaluate students”, represented by normal use cases. In these functionalities there is also an interaction with the agent role, Teacher Tools, that being a reactive agent role, was modeled as an AgentRole_Actor with the stereotype “Reactive_AgentRole” and put within the system boundaries, since it is inserted in the system.

The actor student was associated to the functionalities, “Execute an unmonitored learning session” and “Execute a monitored learning session”, represented as normal use cases. In the first functionality there is also an interaction with the agent role, Unsupervised Learning, that was represented as an AgentRole_Actor with the stereotype “Reactive_AgentRole”.

The functionality, “Execute monitored learning session” involves the five cognitive agent roles, that were classified as PS, UAM and SMI actors, as is demonstrated by their stereotypes in the figure 2. Thus, the SM actor received the UAM_AgentRole stereotype, since it is responsible for the modeling of the students that interact with the system. On its turn the MTP actor received the SMI_AgentRole stereotype, since it is responsible for establishing which methodology will be used with each student. All the other actors received the PS_AgentRole stereotype, because they own knowledge about the problem the application intends to solve.

The SM agent role has for its goal to model the student in a dynamic way. The agent who plays this role needs to perceive when the learning session is beginning and, in this case, trigger the plan, “Apply questionary”, to determine the student profile. He also needs to perceive when the student behavior changes, thereby to trigger the plan to remodel the student profile.

We associated the SM agent role to an IUC with the Goal stereotype representing the goal to which the agent who plays this role has to model the student profile. Note that this goal has two inclusion actionperception (IAP) associations (<<include ap>>) with two IUCs that represent the perceptions the agent needs to own to determine whether it is necessary to trigger any of the plans associated with the goal. Thus, to achieve its goal, the SM agent role has to perform mandatorily these perceptions. So, we used an IUC with the Perception stereotype to represent the perception of the learning session beginning and other IUC with the same stereotype to represent the perception of the student behavior.

In the IUC that represents the goal, there are two Plan Extension Points that represent those points in the goal behavior where the plan associated to it can be extended and
Besides establish also the conditions for those plans to be performed. So, in the moment the agent perceives the learning session is beginning, it will trigger the plan to apply a questionnaire to the student, represented by an IUC with the stereotype Plan; and, if the agent perceives a change in the student behavior, it will forthwith trigger the plan to remodel the student, equally represented by an IUC with the Plan stereotype. Both IUCs that represent the plans are associated with the IUC Goal by means of plan extend associations, i.e., these IUCs only will be performed when the conditions detailed by the Plan extension points are satisfied. Both plans own an action which is also represented by an IUC with the Action stereotype, and it represents the sending of the student model to the agent who plays the MTP agent role.

On its turn, the MTP agent role has for its goal to choose the methodology and the teaching plan most adequate to the student; to do this, it needs to perceive when the student model changes. This goal is associated to a plan, “Change Learning Methodology and Teaching Plan”, that will be triggered whenever the student's model changes or every time the student's performance is proven to be poor. The execution of this plan includes the sending of a message to the agent who plays the LO agent role, to inform the latter that the methodology has been changed.

To model these requirements we associated an IUC with the Goal stereotype with the agent role in order to represent its goal. Next, we associated, by means of IAP associations, two IUCs with the Perception stereotype, to represent the student model change and the student performance perceptions. After that, we created a plan extend association to link the IUC with the Plan stereotype, that represents the plan to change the methodology and the teaching plan, with the IUC Goal. This plan will be triggered only when the student model changes or when the student performance is poor, as is shown by the Plan extension points of the IUC, Goal. Finally, if the plan is triggered, it will be necessary to communicate to the agent who plays the LO agent role the methodology change; as this is done by means of a communication between agents, we identified it as an action and we associated it to the plan by means of an IAP association.

The LO agent role has for its goal to present learning contents for the student and for its perception the choice of the methodology and teaching plan. When the methodology and the teaching plan are perceived, the agent who takes the LO agent role executes the plan “Select material for learning”. To model these requirements we represented the goal, the perception, and the plan as IUCs containing respectively the stereotypes, Goal, Perception and Plan. As the plan will only be triggered when the choice of a methodology is perceived, there is an IAP association between the goal and the perception, forcing the agent to verify whether it occurs. There is also an extension association between the goal and the plan, since the plan only will be triggered if the condition is satisfied.

The LA agent role has for its goal to check the knowledge acquired by the student, represented by an IUC containing the Goal stereotype. To execute this goal, the agent who takes this role must perceive the student's performance; this perception is represented by an IUC containing the Perception stereotype. Besides, the agent who assumes the LA agent role must inform this performance to the agent who takes the MTP agent role, which is represented as an IUC containing the Action stereotype. If the student performance is considered low, then the plan, “Boosting student”, is triggered, equally represented as an IUC with the Plan stereotype. This plan has for action to send motivation messages to the student; we identified this as an IUC containing the Action stereotype and we connected this to the plan by means of an IAP association.

Finally, the KAO agent role, has for its goal to evaluate the student after the session ends. Thus, it needs to perceive that the learning session has ended so as to know when to trigger the plan, “Apply evaluation”.

VII. Validating the Proposed Metamodel

With relation to the validation, we compared the models created by means of this metamodel with the models created from the UML original UCD, trying to demonstrate that the proposed metamodel allows the modeling of concepts not supported by the standard UCD. Following this idea, we produced a UCD for the AME-A system using only the standard UML and comparing it with the diagram produced by means of our metamodel, as can be seen in the next figure.

![Figure 3. Use-Cases Diagram Using the Standard UML](image-url)

We did not consider possible to model the goals, plans, perceptions, or actions of these roles, because they are internal functionalities to which the external users do not have access. Besides, the standard UML simply does not have mechanisms to represent goals, plans, perceptions, or actions in a UCD.

Alternatively, using the standard UML, we could try to make the use case “Execute monitored learning session”, to encompass the steps of the use cases, “Student modeling”, “Choose methodology and teaching plan”, “Present material for learning”, “Verify student knowledge”, and “Evaluate student”. What could be done in this specific situation, would be to associate the use cases that represent the plans to the use case, “Execute a monitored learning session”, by means of extend associations, establishing the conditions for those use cases to be performed, as demonstrated in the figure 4.

By this approach we tried to use the standard UML in order to achieve the same objective reached by the use of the proposed metamodel. As can be perceived we kept suppressing the representation of agents, as these are internal to the system.
and an actor represents entities external to the system.

![Diagram of AME-A](image)

Figure 4. An alternate for the AME-A modeling using extend associations

We tried to associate to the use case, “Execute a monitored learning session”, the use cases originally used to represent the plans that should be performed by the agents. Thus, we attributed the plan extension points of the Goal internal use cases to the use case, “Execute a monitored learning session”, thus trying to demonstrate under which conditions each “plan” could be performed.

However, these functionalities were not conceived as services to be performed in the “traditional” way by the software but to be divided among cognitive agents, i.e., they are functions separated from the “Execute a monitored learning session” functionality, that represents a service offered to the student, where he will choose, for instance, the learning session theme, but thereafter there will be an interaction with the agents, when those will assume a series of tasks to aid the student and to do it they will need to perceive events, take decisions, and perform actions with respect to which attitudes should be performed. It is not possible to render explicit these kinds of functionalities by using the standard UML.

It is important to note that in reality the use cases associated by means of extensions are not functionalities that the student actor has access to; these functionalities should be performed by agents according to their goals. We further highlight that it is not possible to represent the goals of an agent nor the plans associated to each goal using the standard UML. Likewise, the UML does not support the perception and action representation nor it is able to represent internal agents as actors.

VIII. CONCLUSIONS

We presented a UML metamodel developed for the requirements analysis for the multiagent systems project. This metamodel allows to represent agent roles, besides modeling the perceptions and actions the roles should own, as well as the goals of these roles, together with the plans to achieve them and the conditions for these plans to be performed. We demonstrated the applicability of this metamodel by means of the AME-A intelligent tutor system modeling.

The metamodel in question fills a gap not focused by other approaches that extended the UML language for the MAS project, since none of the studied languages was concerned with addressing the requirements analysis question, a phase of extreme importance for the success of a software project.

With respect to the metamodel validation, we compared the models created by means of this profile with models created from the UML original use-cases diagram; we thus tried to demonstrate that the new proposed metamodel allows to model the concepts not supported by the standard use-cases diagram.

REFERENCES