A UML Profile Oriented to the Requirements Modeling in Intelligent Tutoring Systems Projects

Gilleanes Thorwald Araujo Guedes and 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
gtaguedes@inf.ufrgs.br, rosa@inf.ufrgs.br
http://www.inf.ufrgs.br/pos

Abstract. This paper describes a proposal for the creation of a UML profile oriented to the intelligent tutoring systems project. In this paper we shall describe the proposed profile as well as its application into the modeling of the AMEA intelligent tutoring system.

Keywords: UML Profiles, Stereotypes, Actors, Use-Cases, Agents, AMEA.

1 Introduction

In the area of Artificial Intelligence, the employment of intelligent agents as auxiliary aids to software applied to the most diverse dominions is being spread out. This practice has shown to be a good alternative for the development of complex systems, fostering a great increase of agent-supported software development in the several areas, one of those the intelligent tutoring systems, in which agents are also employed as pedagogical agents.

However, the development of this kind of system has presented new challenges to the software engineering area and this led to the surfacing of a new sub-area, blending together concepts brought over from both the software engineering and artificial intelligence areas, which is known as the AOSE - Agent-Oriented Software Engineering, whose goal is that of proposing methods and languages for projecting and modeling agent-supported software.

Despite many AOSE methods having been created for multi-agent systems (MAS) projects, Vicari [1] states these are not totally adequate for the modeling of Intelligent Tutoring Systems. Likewise, some attempts to adapt and extend UML (Unified Modeling Language) to the multi-agent systems projects were made; nonetheless, those attempts we have studied did not concern themselves into extending and applying some of the UML resources, such as the use-case diagram, which is mainly employed for requirements collecting and analyzing, an essential phase for the achievement of a good system project. Thus, we developed a UML profile for the project of Intelligent Tutoring Systems, for which end we started by adapting the use-case diagram.

M. Bramer (Ed.): IFIP AI 2010, IFIP AICT 331, pp. 133–142, 2010.

2 UML, Metamodels and Profiles

According to [2], the UML is a visual language for specifying, constructing, and documenting the artifacts of systems. It is a general-purpose modeling language that can be applied to all application domains.

The UML specification is defined using a meta-modeling approach (i.e., a meta-model is used to specify the model that comprises UML) that adapts formal specification techniques. When meta-modeling, we initially establish a distinction between meta-models and models. A model typically contains model elements. These are created by instantiating model elements from a meta-model, i.e., meta-model elements. The typical role of a meta-model is that of defining the semantics for the way model elements within a model get instantiated.

A Profile is a kind of Package that extends a reference meta-model. The primary extension construct is the Stereotype, which is defined as part of the Profiles. A profile introduces several constraints, or restrictions, on ordinary meta-modeling through the use of the meta-classes defined in this package.

3 UML-Derived Languages

Some attempts have already been tried to adapt UML for the project of multi-agent systems, though nothing specific for the project of intelligent tutor systems. One of the first attempts was the AUML language [3]. Besides that, other languages, like AML [4], AORML [5], and MAS-ML [6] were also proposed.

However, neither of the above focus the matter of requirements collecting and analyzing nor on its modeling by means of the UML use cases diagram and no attempt was found in those languages to extend the metaclasses employed in that diagram for applying them on the multi-agent systems project.

4 UML Profile for the ITS Project

Considering that UML is a standard modeling language broadly accepted and understood in the software engineering area, that multi-agent systems own their proper characteristics, and that very few of works applying UML into a multi-agent systems project did care to focus the matter of requirements collecting and analyzing, we decided on creating a UML profile in which we extend the meta-classes employed by the use-case diagram, thus creating stereotypes prepared to identify the particular functionalities of this kind of system, as can be seen in the following illustration.

To develop this profile, we began by using the original metaclass, Actor. According to [7], the Actor metaclass represents a type of role played by an entity that interacts with the subject, but which is external to the subject. An actor does not necessarily represent a specific physical entity but merely a particular facet (i.e., "role") of some entity that is relevant to the specification of its associated use cases.

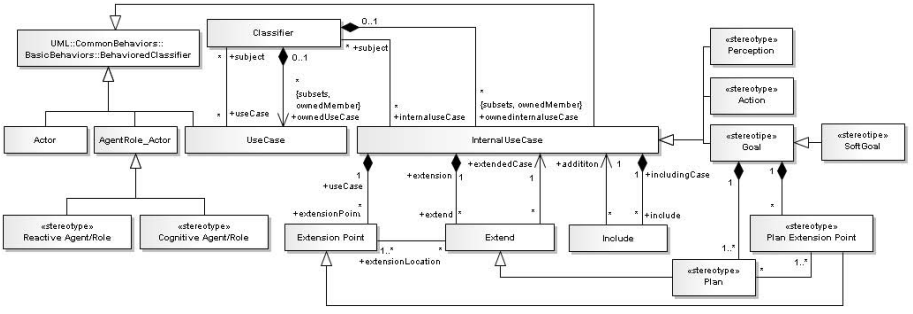

Fig. 1. UML Profile for the Project of Intelligent Tutoring Systems

However, most of the times, the software agents are not external to the software, rather they customarily are inserted in the system environment and, as they are independent, proactive, and able to interact with the software according to their goals, these should be represented as actors. So, as states [8], it is necessary to adapt this concept, considering that agents can be internal to the system, that is, an agent can be a part of the subject and, therefore, if we are to represent agents (their roles) as actors, said actors should be internally represented within the system's borders, for they belong to the software.

The representation of agents/roles as UML actors can also be seen in [9], in which agents represent active objects and are modeled within the system as rectangle-headed actors. However [8] only suggests that the concept of actor should be adapted and [9] did not created a UML profile for agents. In our work we explicitly derived new metaclasses from the original metaclasses endeavoring for the former to allow an adequate requirements modeling in MAS.

However, according to [2], it is not possible to take away any of the constraints that were applied to a metamodel such as UML when using a profile. Thus, instead of extending the Actor metaclass, we created a new metaclass derived from the same metaclass as the Actor metaclass had been, as can be seen in figure 1, creating the Agent/Role-Actor Metaclass. In this new metaclass we copied all the characteristics of the Actor metaclass, including the same symbol, suppressing only the constraint that an actor needs must be external to the system. And, from this new metaclass we derived the Reactive Agent/Role and Cognitive Agent/Role metaclasses.

As can be observed, we applied the stereotype "stereotype" in both metaclasses, which means that these metaclasses will be applied as stereotypes on agent/role actors while attributing them special characteristics, for said agent/role actors shall be utilized within specific domains to represent cognitive and reactive agents/roles. We opted for creating two stereotypes so as to establish a difference between reactive and cognitive agents, considering that either group presents different characteristics.

According to [7], a use case is the specification of a set of actions performed by a system. Each use case specifies some behavior that the subject can perform

in collaboration with one or more actors. Use cases can be used both for specification of the (external) requirements on a subject and for the specification of the functionality offered by a subject. Moreover, the use cases also state the requirements the specified subject poses on its environment by defining how they should interact with the subject so that it will be able to perform its services.

Thus, with the objective of adapting the use cases concept to the MAS modeling, we derived a new metaclass departing from the same metaclass as the UseCase metaclass, called "InternalUseCase" and then we created the same relationships with the Classifier metaclass for this new metaclass as those belonging to the UseCase metaclass. We made it that way because we intended to represent goals, plans, actions and perceptions as use cases, but the semantics of the UseCase Metaclass says that use cases represent external requirements, therefore, to adapt this concept to the MAS modeling we created a similar metaclass, all the while we modified its semantics to represent internal requirements.

Naturally, all Internal use cases are internal to the system and can be neither used nor seen by human users. From the InternalUseCase metaclass, we extended some metaclasses to attribute special characteristics to internal use cases; these extended metaclasses will be employed as stereotypes. Thusly, we created the metaclasses, Perception and Action, to model internal use cases that contained the necessary steps for an agent to perceive or do something, a procedure also suggested by [8] to be applied in normal use cases, though as stated before in [7], these refer to the external requirements, not to the internal ones.

A third metaclass was derived from the metaclass, InternalUseCase, to represent Goals. A internal use case employing the stereotype, Goal, shall contain a description of a desire to be attained by an agent and the possible conditions for that desire to become an intention. Besides, we evolved the first proposal of this profile presented in [10] when we included the concept of softgoal, deriving the SoftGoal metaclass from the Goal metaclass. The concept of softgoal is described in [11] and it is used to represent cloudy goals, which do not present a clear definition and/or criteria for deciding whether they are satisfied or not.

A somewhat similar proposal for representing goals as use cases can be seen in [9], but, besides using internal use cases, we went a little further beyond the proposal presented by [9], when we considered that, just like a goal represents a desire that will not necessarily become an intention, the steps for its execution should be detailed in other or others internal use cases; in the situation of a internal use case employing the stereotype, Goal, we shall only detail those perceptions and conditions necessary for that goal to become an intention. Considering a goal might eventually have more than a plan and that this or these plans will only be accomplished within certain conditions, we decided on deriving still another metaclass, Plan, from the metaclass, Extend.

According to [7] a metaclass, Extend, represents a relationship from an extending use case to an extended use case that specifies how and when the behavior defined in the extending use case can be inserted into the behavior defined in the extended use case. If the condition of the extension is true at the time the first extension point is reached during the execution of the extended use case, then

all of the appropriate behavior fragments of the extending use case will also be executed. If the condition is false, the extension will not occur. Because a plan only will be triggered after some particular condition be satisfied, we decided on extending the metaclass, Plan, from the metaclass, Extend, and associated the former to the metaclass, Goal, by means of a composite association.

Finally, we derived the meta-class, Plan Extension Point from the meta-class, Extension Point. According to [7] an extension point identifies a point in the behavior of a use case where that behavior can be extended by the behavior of some other (extending) use case, as specified by an extend relationship. We derived this last meta-class only to set up a difference between Plan Extension Points and normal Extension Points; nonetheless, this also can serve to render explicit the condition for a plan to be executed.

5 The AMEA Project

The AME-A [12] architecture - Ambiente Multi-Agente de Ensino-Aprendizagem or "Teaching-Learning Multi-agent Environment" is composed by a hybrid society of agents that cooperate to aid into the students learning process. The environment interacts with human agents that can be both the teacher or the students and owns several reactive and cognitive agents.

The teacher can create a learning activity or evaluate the students with the help of the Teacher's Tools reactive agent. The student, on his turn, can choose between execute an unmonitored learning session or a monitored learning session. In the first option, the student only interacts with the Unsupervised Learning reactive agent that will only present him/her the contents to be learned.

The monitored learning activity is set as the main focus for the system, in which it aims to maximize the student learning by means of the aid of five cognitive agents, 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 learning plan that are more adequated to the student profile every time it changes or whenever the student performance is lower than the expected level; the LO agent selects the contents to be taught and the way how these will be presented according to the chosen methodology; the LA agent checks on the student performance throughout the session and the KAO agent applies a general evaluation after the session ends.

Since the teacher and the student are human agents that interacts with the system but are external to it, we represent them as normal actors, out of the system boundaries and we associate to the actor that represents the teacher to the funcionalities represented by normal use cases, "Create learning activity" and "Evaluate students". Notice that in these funcionalities there is also an interaction with the Teacher Tools agent, who, being a reactive agent, was represented as an agent/role actor with the stereotype "Reactive Agent/Role" and placed inside the system boundaries, because it is inserted in the software.

Now the actor student was associated to the funcionalities "Execute an unmonitored learning session" and "Execute a monitored learning session", equally represented as normal use cases. In the first funcionality there is also a interaction with the reactive agent Unsupervised Learning, which, in the same way as the previous agent was represented as an agent/role actor with the estereotype "Reactive Agent/Role".

The funcionality "Execute monitored learning session" represents the more important activity of the system and it is the one that presents more complexity, involving the five cognitive agents of the software, which, because are agents, are represented as Agent/Role Actors inside of the system, containing the "Cognitive Agent/Role" stereotype, since they are cognitive agents.

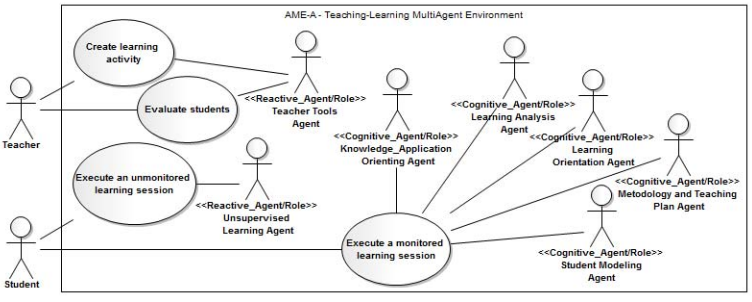

Fig. 2. Normal Actors modeling external users, normal Use Cases modeling functionalities, and stereotyped Agent/Role Actors modeling agents

These funcionalities were represented as normal use cases, since they represent services that the external users can ask from the system. Figure 2 ilustrates this excerpt of the modeling. This figure presents only a part of the use cases diagram. Due to the lack of space and for a matter of visibility, we suppressed the representation of the cognitive agents goals, plans, perceptions and, actions, detailing separately the requirements for each cognitive agent. To sustain the connection with this excerpt of the modeling, we shall mantain the association of the cognitive agents with the use case "Execute monitored learning activity".

The SM agent has for its goal modeling the student in a dynamic way. This agent has two perceptions. First it must perceive that the learning session beginning and, in this case, trigger the "apply questionary" plan to determine the student profile. And it needs perceive when the student behavior changes, in which case it has to trigger the plan to remodel the student profile. Figure 3 ilustrates the modeling of these requirements for the SM agent.

In this figure we associated to the SM agent an internal use case (IUC) with the Goal stereotype representing the goal that this agent has to model the student profile. This goal has two inclusion associations with two IUCs that represent the perceptions the agent needs to own to determine wether is necessary to trigger some of the plans associated to the goal. An inclusion association determines that

the behavior of the use case therein included should mandatorily be executed by the use case it includes. So, to reach its goal, the SM agent has to execute these perceptions. So that, we use an IUC with the stereotype Perception to represent the perception of learning session beginning and another IUC with the same stereotype to represent the perception of the student behavior.

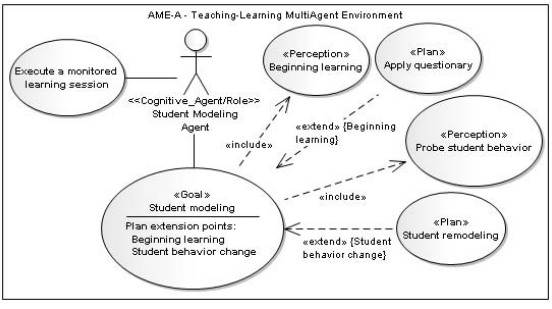

Fig. 3. Internal use cases with Goal, Plan, and Perception stereotypes

Notice that, in the IUC that represents the goal, there are two Plan extension points, both representing the points in the goal behavior to where the plans associated to it can be extended, besides establishing the conditions for the plans to be executed. In such way, if the agent perceives the learning session is beginning, the plan to apply a questionnaire to the student will be triggered, represented by an IUC containing the Plan stereotype; and, if the agent notices a change in the student behavior, the plan to remodel the student will be triggered, equally represented by an IUC containing the Plan stereotype. Further observe that the two IUCs that represent the plans are associated to the IUC Goal by means of extension associations, that is, these IUCs will only be executed after the conditions detailed by the Plan extension points are satisfied."

The following agent, MTP, has for its goal to choose the methodology and teaching plan which is more adequated to the student, to do so, it has to perceive when the student model changes, this includes the perception of when the student is modeled for the first time, when there is no model yet. This goal han an associated plan, "Change Learning Methodology and Teaching Plan" which will be triggered when the studel model changes or when the student's performance is low. The execution of this plan includes the sending of a message to the LO agent informing that the methodology was modified.

In figure 4, to model these requirements we associated an IUC including the Goal stereotype with the agent to represent its goal. After that, we associated, by means of inclusions, two IUCs with the Perception stereotype to represent the perceptions of student model change and student performance. After that, we created an extension association to connect the IUC with the Plan stereotype, which will represent the plan for methodology and teaching plan changes to the Goal IUC. This plan will be triggered only when the student model changes or

when the student performace is low, as demonstraded by the Plan extension points in the Goal IUC. Finally, if the plan is executed, it is need to comunicate to the LO agent the chage of methodology; since this is done by means of a communication between agents, we identified this as an action and we associated it to the plan by means of an IUC with the Action stereotype.

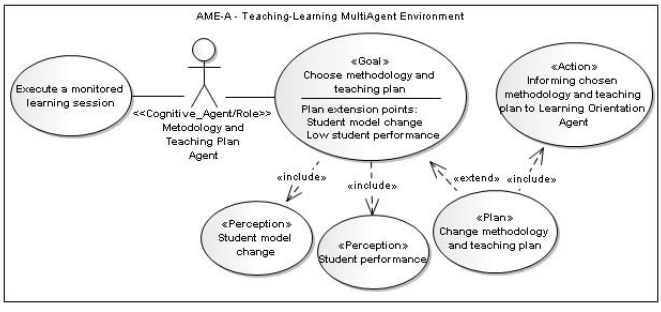

Fig. 4. Methodology and Teaching Plan agent requirements

The LO agent 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 perceiveds by the LO agent, it executes the plan "Select material for learning". These requirements are modeled on figure 5, where we represent the goal, the perception and the plan as internal use cases 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 inclusion association between the goal and the perception, obligating the agent to verify if 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.

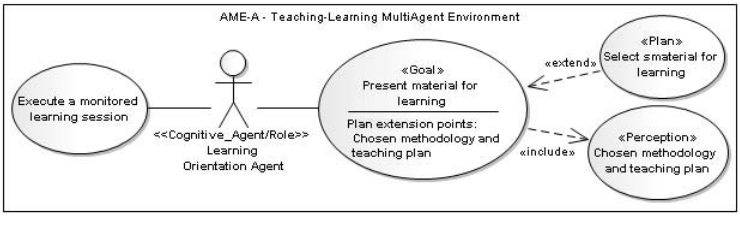

Fig. 5. Learning Orientation agent requirements

Figure 6 presents the modeling related to the requirements of LA agent, which 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 must perceiving the student's performance; this perception is represented by an IUC

containing the Perception stereotype. Besides, the LA agent must inform this performance to the MTP agent, which is represented as an IUC containing the Action stereotype. If the student performance is considerated 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 plan by means of an inclusion association.

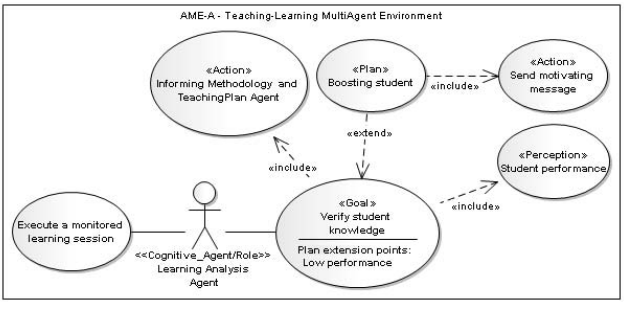

Fig. 6. Learning Analysis agent requirements

The last agent involved in the process, the KAO agent, has for its goal to evaluate the student after the session ends. Thus, it needs to perceive the learning session ends so as to know when to trigger the plan "Apply evaluation". Here we applied the same stereotypes used in the previous examples in internal use cases, as demonstrated in figure 7.

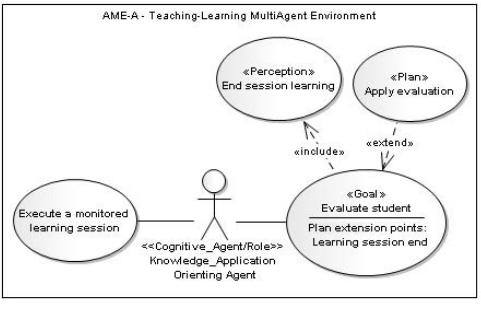

Fig. 7. Knowledge Application Orienting agent requirements

There are structural and behavioral questions that need to be best detailed, since the presented profile focus only the system requirements. All the same, even on project level, it is needed to represent information like beliefs, desires and intentions by means of class diagrams and the processes represented by the IUCs must be detailed through behavioral diagrams. Up to now we have been

directly using these diagrams in [10], though we might instead extend other UML metaclasses so as to best adequate them to the ITS Project or we can apply some of the UML derived languages to the MAS project previously described.

6 Conclusions

Throughout this paper we have presented a UML profile developed to the intelligent tutoring systems project oriented to the requirements collection and analysis. We demonstrated the applicability of said profile by means of AMEA System modeling and also that the stereotypes we have created can be used to model cognitive and reactive agents, and actions, perceptions, goals, and plans as well. However, we intend to apply this profile on some other projects for Intelligent Turoring Systems, possibly of a more complex sort, so as to find out whether this profile is really adequate for projecting this type of systems and to pinpoint any weakness needing some improvement.

Although this profile has been developed for the Intelligent Tutoring Systems modelling,we believe it might be applied to other MultiAgent System projects oriented to other dominions. We also believe this profile can be adapted to most UML already existing extensions created for the MultiAgent Systems project.

References

1. Vicari, R.M., Gluz, J.C.: An Intelligent Tutoring System (ITS) View on AOSE. International Journal of Agent-Oriented Software Engineering (2007)
2. OMG - Object Management Group. Unified Modeling Language: Infrastructure Specification - Version 2.1.1. OMG (2007), `http://www.omg.org`
3. AUML Official Website (2007), `http://www.auml.org`
4. Trencansky, I., Cervenka, R.: Agent Modeling Language (AML): A comprehensive approach to modeling MAS. Informatica 29(4), 391–400 (2005)
5. Wagner, G.: A UML Profile for External AOR Models. In: Third International Workshop on Agent-Oriented Software Engineering (2002)
6. Silva, V., et al.: MAS-ML: A Multi-Agent System Modeling Language. International Journal of Agent-Oriented Software Engineering, Special Issue on Modeling Languages for Agent Systems, Inderscience Publishers 2(4) (2008)
7. OMG - Object Management Group. Unified Modeling Language: Superstructure Specification - Version 2.1.1. OMG (2007), `http://www.omg.org`
8. Bauer, B., Odell, J.: UML 2.0 and Agents: How to Build Agent-based Systems with the new UML Standard. Journal of Engineering Applications of AI (2005)
9. Flake, S., Geiger, C., Küster, J.: Towards UML-based Analysis and Design of Multi-Agent Systems. In: Proceedings of ENAIS 2001, Dubai (March 2001)
10. Guedes, G.T.A., Vicari, R.M.: A UML Profile Oriented to the Requirements Collecting and Analyzing for the MultiAgent Systems Project. In: Proceedings of SEKE 2010, San Francisco Bay (2010)
11. Bresciani, P., et al.: Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems 8(3), 203–236 (2004)
12. D'Amico, C.B., Viccari, R.: Adapting Teaching Strategies in a Learning Environment on WWW. WebNet (1998)