EXTENDING MAS-ML TO MODEL PROACTIVE AND REACTIVE SOTWARE AGENTS

Enyo José Tavares Gonçalves
Instituto Federal de Educação Ciência e Tecnologia do Ceará, Maracanaú, CE, Brazil

Mariela I. Cortés, Gustavo A. L. de Campos
Universidade Estadual do Ceará, Fortaleza, CE, Brazil

Viviane Torres da Silva
Universdade Federal Fluminense, Niterói, RJ, Brazil

Keywords: Proactive and Reactive Agents, Multi-agent System, Modelling Language, Conceptual Framework.

Abstract: The existence of Multi Agent System (MAS) where agents with different internal architectures interact to achieve their goals promotes the need for a language capable of modeling these applications. In this context we highlight MAS-ML, a MAS modeling language that performs a conservative extension of UML while incorporating agent-related concepts. Nevertheless MAS-ML was developed to support pro-active agents. This paper aims to extend MAS-ML to support the modelling of not only proactive but also reactive agents based on the architectures described in the literature.

1 INTRODUCTION

Nowadays, the agent technology has been widely applied to solve a vast set of problems. Russell and Norvig (2003) define an agent as an entity that can perceive its environment through sensors and act in environment through actuators. Unlike objects, agents are more complex entities with behavioural properties, such as: (i) they are autonomous and not passive, and (ii) able to interact through exchange of messages and not by explicit task invocation (Wagner, 2003). Multi-Agent System (MAS) is the sub-area of Artificial Intelligence that investigates the behaviour of a set of autonomous agents, aiming to resolve a problem that is beyond the capacity of a single agent (Jennings, 1996).

The agent-oriented development paradigm requires adequate techniques to explore its benefits and features, in order to support the construction and maintenance of this type of software (Zambonelli et al., 2001). A simple agent is classified according to its internal architecture that determines distinct agency properties, attributes and mental components. These features introduce additional complexity to the system development. A MAS may encompass multiple types of agents with different internal architectures (Weiss, 1999). Thus, the existence of a language to support the modelling of different internal agent architectures is strongly desirable

Several modelling languages have been proposed in the literature to model agents and their systems. One of them is called MAS-ML (Multi-Agent System Modelling Language) (Silva and Lucena, 2004) (Silva, Choren and Lucena, 2008a) that performs a conservative extension to UML based on the agent-oriented concepts defined in the conceptual framework TAO (Taming Agents and Objects) (Silva and Lucena, 2004). In particular, the following characteristics of the language can be highlighted: (i) the support for the modelling of main MAS entities: agents, organization and environments; (ii) the support for conventional objects; (iii) the support for modelling static and dynamic properties; (iv) the modelling of agent roles, that are important while defining agent societies; and (v) the clear justified extension of the UML metamodel to model agent-related properties based on TAO (Silva, Choren and Lucena, 2008a). Due to its characteristics, MAS-ML is known as one of the main adequate modelling language to model MAS.

José Tavares Gonçalves E., I. Cortés M., A. L. de Campos G. and Torres da Silva V. (2010).
EXTENDING MAS-ML TO MODEL PROACTIVE AND REACTIVE SOTWARE AGENTS.
In *Proceedings of the 12th International Conference on Enterprise Information Systems - Artificial Intelligence and Decision Support Systems*, pages 75-84
DOI: 10.5220/0002907700750084
Copyright © SciTePress

MAS-ML was originally designed to support the modelling of only proactive agents that are goal-oriented entities and guided by pre-established plans. However, not all MAS require or permit their agents are pro-active, as the case of simulations for an ant colony (Dorigo and Stützle, 2004). Create goal-based agents with plan in stochastic and partially observable environments can be a very complex task (Weiss, 1999). Therefore, it is fundamental to extend MAS-ML to be able to model not only proactive agents that have pre-defined plans but also reactive ones. In addition, MAS-ML should also be able to model proactive agents able to create new plans and that use utility functions to execute (Russell and Norvig, 2003).

In this paper, we describe an extension of the MAS-ML in order to capture the reactive agents, proactive agents with planning and proactive agents based on utility functions. The paper is structured as follows. The main internal architectures for agents are described in Section 2. Section 3 briefly presents MAS-ML modelling language. The extension of MAS-ML is then detailed in Section 4. In Section 5 the modelling of the TAC-SCM (Trading Agent Competition - Supply Chain Management) (Sadeh et al., 2003) application is presented by using the extended MAS-ML. Related works are described in Section 6 and, finally, conclusions and future works are discussed in Section 7.

2 AGENT ARCHITECTURES

The agent internal architectures can be categorized based on proactive and reactive foundations.

2.1 Simple Reflex Agents

A simple reflex (or reactive) agent (Russell and Norvig, 2003), is considered the most simple internal architecture. Condition-action rules are used to select the actions based on the current perception. These rules follow the form: “if condition then action”, and determine the action to be executed if the perception occurs. This architecture assumes that at any time the agent receives information from the environment though sensors. These perceptions consist of the representation of state aspects that are used by the agent for making decision. A subsystem is the one responsible for the making decisions, i.e., responsible for processing the perception sequence and selecting a sequence of actions from the set of possible actions for the agent. The agent performs the selected action upon an environment through actuators.

2.2 Model-based Reflex Agents

The structure of this kind of agent is similar to the simple reactive agent presented before since it deals with the information by using condition-action rules. In order to handle partially observable environment and to reach a more rational performance, the agent is able to store its current state in an internal model.

According to Weiss (1999), reflex agents with internal states select actions by using the information in its internal states. A function called *next function* is introduced to map the perceptions and the current internal state into a new internal state used to select the next action. Such state describes aspects of the world (called model) that cannot be seen in the current moment, but it was perceived previously or has come out by inferences (Russell and Norvig, 2003).

2.3 Goal-based Agents

Sometimes, the knowledge about the current state of the environment is not enough to determine the next action and additional information about desirable situations is required. Goal-based agents are model-based agents that set a specific goal and select the actions that lead to that goal. This allows the agent to choose a goal state among multiple possibilities.

Planning activity is devoted to find the sequence of actions that are able to achieve the agent's goals (Russell and Norvig, 2003). The sequence of actions previously established leads the agent to reach a goal is termed plan (Silva, and Lucena, 2004) (Silva, Choren and Lucena, 2008a). Thus, the goal-based agent with planning involves the *next function* component and also includes the following elements:

- *Formulate Goal Function*, which receives the state and returns the formulated goal.
- *Formulate Problem Function*, which receives the state and the goal and returns the problem.
- *Planning*, that receives the problem and uses search and/or logic approach to find a sequence of actions to achieve a goal.
- *Action* that is represented with its pre-conditions and post-conditions.

2.4 Utility-based Agents

Considering the existence of multiple goal states, it is possible to define a measure of how desirable a particular state is. In this case, aiming to optimize the agent performance, the *utility function* is responsible for mapping a possible state (or group of states) to a measure of utility associated, according to the current

goals (Russell and Norvig, 2003). Thus, the utility function is incorporated into the architecture.

In addition, the utility-based agent preserves the same elements that a goal-based agent: *Next function, formulate goal function, formulate problem function, planning and action*.

3 MAS-ML

MAS-ML was originally designed to support the modelling of proactive agents that are goal-based and guide by pre-established plans.

MAS-ML model all structural and dynamic aspects defined in TAO metamodel by extending the UML metamodel. The structural diagrams defined by MAS-ML are Role Diagram, Class Diagram and Organization diagram (Silva, Choren and Lucena, 2004). Using the three static diagrams is possible model all structural aspects of entities defined in TAO. The main element of the agent-oriented modelling is the *agent* itself. Figure 1 shows the diagram element used in static diagrams of MAS-ML to represents agents.

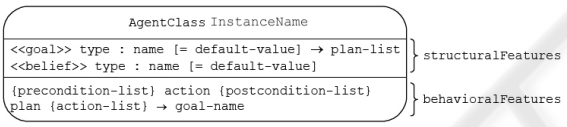

Figure 1: An instance of AgentClass metaclasse.

The dynamic diagrams defined in MAS-ML are extended versions of the UML Sequence Diagram and Activities Diagram (Silva, Lucena and Choren, 2008b).

4 MAS-ML EXTENSIONS

This section presents the extension to MAS-ML in order to support the modelling of agents by using diverse internal architectures: Simple reflex, Model-based reflex, Goal-based and Utility-based. The new version of MAS-ML is named of MAS-ML 2.0.

According to UML (2009), tagged values, stereotypes and constraints are extension mechanisms. Additionally, adaptation of existing metaclasses and definition of new metaclasses can also be used. Stereotypes and definition of new metaclasses was used to represent Simple reflex agents, Model-based reflex agents, Goal-based agents with planning and Utility-based agents. Following the architecture definitions presented in Section 2, the characteristics that need to be defined are Perception, Next-function, Formulate-goal-function, Formulate-problem-function, Planning and Utility-function. The Figure 2 illustrates the MAS-ML metamodel extensions.

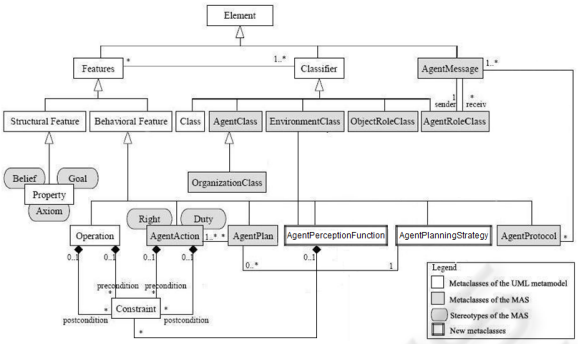

Figure 2: MAS-ML metamodel extension.

The Perception collects information about the environment and/or other agents, without modify them. Since there is not any metaclasse in MAS-ML that can be used to represent such concept, the *AgentPerceptionFunction* metaclass was created to represent the agent perception.

The agent perceptions can be also represented on the environment, since it represents the elements that the agent can perceives and as far as the sensors of the agent will perceives (partially or fully, for example). Once the environment influences the perception of the therein agents, an association was established between the metaclasses AgentPerceptionFunction and EnvironmentClass.

The Planning task results in a sequence of actions in order to achieve a goal (Russell and Norvig, 2003). In addition, the following properties are observed: (i) unlike a plan (represented by *AgentPlan* in the original MAS-ML metamodel), the sequence of actions is created at runtime; and (ii) unlike a simple action (represented by *AgentAction* in the MAS-ML metamodel), the action of planning has a goal associated. Thus, the new metaclass *AgentPlanningStrategy* was created to represent the planning. An association relationship between *AgentPlanningStrategy* and *AgentPlan* was defined to represent that the action of planning can create plans.

The metaclasses *AgentPerceptionFunction* and *AgentPlanningStrategy* extends the *BehavioralFeature* metaclass. The *AgentPerceptionFunction* has a *Constraint* that is used to restrict the information that can be perceived through the agent sensors.

The *Next-function, Formulate-goal-function, Formulate-problem-function* and *Utility-function* are special agent actions that depend on the agent internal architecture. The <<next-function>>, <<formulate-goal-function>>, <<formulate-problem-function>> and

<<utility-function>> stereotypes was thus created and related to *AgentAction* metaclass.

Finally, the condition action rules of reactive agents, can be represented by using the agent's action representation (Silva, Choren and Lucena, 2008a), which may have a pre-condition attached.

4.1 Static Representation of AgentClass

The new structural and behavioural features in the modelling of the different types of agents influence the AgentClass metaclass representation in static diagrams.

4.1.1 Simple Reflex Agent Structure

The representation for a Simple Reflex Agent (Figure 13) does not include any structural element since neither goals nor beliefs are inherent to this architecture. In the lower compartment the perceptions and actions, driven by condition-action rules and not by a specific plan, are represented.

4.1.2 Model-based Reflex Agent Structure

The model-based reflex agents represent an upgrade over the simple reflex agents. Thus, the definition for the action element is kept the same. In addition, beliefs representing the state and the next function are included. Figure 12 presents the graphical representation of AgentClass for a Model-based reflex agent.

4.1.3 Goal-based Agent with Plan Structure

The goal-based agents with plan have the same structure proposed initially by Silva and Lucena (2004) including goals, beliefs, actions and plan. Figure 1 and Figure 11 show the graphical representation of this agent.

4.1.4 Goal-based Agent with Planning Structure

The goal-based agents with planning incorporate additional complexity to the agent representation. Firstly, goals are considered in order to guide the agent behaviour. Thus, this element is included as a structural component denoted with the <<goal>> stereotype. In order to manipulate consistently goals and states, the agent behaviour is enhanced with <<perceives>>, <<formulate-goal-function>> and <<formulate-problem-function>> elements. The already existent <<next-function>> element is keep up. This function receives the current perception and the beliefs that must be updated (state).

In addition, instead of representing pre-established plans, the planning activity is incorporated. This activity involves a goal and uses the available actions to create a sequence of actions. Figure 14 illustrates the AgentClass for a goal-based agent using planning.

4.1.5 Utility-based Agent Structure

The representation for the utility-based agent consists in a specialization of the goal-based agent with planning. However, the <<utility-function>> element is added to represent the function responsible for the optimization of the gent performance. The graphical representation of AgentClass for a pro-active agent based on utility is illustrated in Figure 15.

Along the planning, the agents may be linked to reach more than one goal. In this case, the occurrence of conflicting goals or the existence of several states meeting the goals is possible. So the utility function is incorporated into the agent structure, in order to evaluate the usefulness degree of the associated goals.

4.2 AgentRoleClass Static Representation

An AgentRoleClass in MAS-ML is represented by a solid rectangle with a curve at the bottom. Similarly to the class representation, it has three compartments separated by horizontal lines. The upper compartment contains the agent role name unique in its namespace. The intermediate compartment contains a list of goals and beliefs associated with the role, and below, a list of duties, rights and protocols.

Reactive agents have not explicit goals and, more particularly, the simple reflex agents do not have beliefs. Thus, their role representation must be adapted (Figure 16). In addition to the representation of the roles of simple reflex agents, roles for model-based reflex agents include beliefs in order to partially handle observable environments. The agent role representation in this case is represented in Figure 17.

The features of agent roles in other architectures are inaltered since both define beliefs and goals. The structural changes regarding the AgentRoleClass entity impact the Organization and Roles diagrams.

4.3 Dynamic representation of AgentClass

Similarly to the static diagrams, the new representation of the AgentClass influences the representation of their behavioural features. In follows, the dynamic representation of the different agent types is illustrated through sequence diagrams using MAS-ML 2.0.

4.3.1 Reflex Agents Representation

The agent's perception is represented in sequence diagram of MAS-ML by an arrow with an open mind leaving the agent to the environment, together of the <<perceives>> stereotype, the perception name and the elements that agent can see (Figure 3).

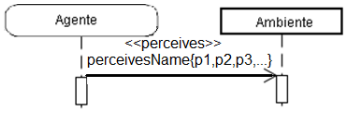

Figure 3: Perception of the agent in sequence diagram.

The actions sequence of the reactive agents cannot be defined in the analysis phase, but it is possible to represent the set of actions with the condition or conditions associated. Figure 4 illustrates the action taken by a reactive agent.

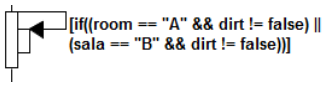

Figure 4: Reactive agent action in the sequence diagram.

The next function is represented in the sequence diagram of MAS-ML 2.0 by a closed arrow with full head, which starts at the agent and ends at the agent, then the stereotype <<next-function>> followed by the name of the function. Figure 5 illustrates the next function in the sequence diagram.

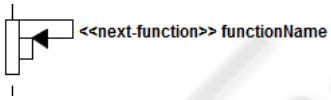

Figure 5: Sequence diagram of the next function.

Therefore, if a simple reflex agent is modelled initially have their perception and then their actions guided by the condition-action rules. In the case of a Model-based reflex agent, initially we have the perception, then the next function and, finally, its actions guided by the condition-action rules.

4.3.2 Proactive Agent Representation

The next function is executed before the formulate goal function and is used by two types of pro-active agents in this paper. The next function element is represented in the sequence diagram by a arrow full head, which begins and ends on the agent in itself, together with the stereotype <<next-function>> and the function name as shown in Figure 5. Then we have the representation of the formulate problem function in the sequence diagram. The representation is done by an arrow full head, which begins in the agent and ends in itself, accompanied by the relevant stereotype. The figures 6 and 7 illustrate the formulate goal function and the formulate problem function, respectively.

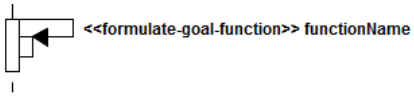

Figure 6: Formulate goal function in sequence diagram.

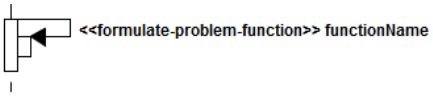

Figure 7: Formulate problem function in sequence diagram.

In case of agents that use planning, the actions sequence that the agent will take to achieve the goal cannot be advanced before its execution. In this case, planning is represented by a closed arrow head that begins and ends in the agent in itself accompanied by the stereotype <<planning>>. The actions that can be used for planning to achieve (s) objective (s) are represented as initially defined by Silva (2004). Optionally, the criterion or algorithm used to perform the planning can be specified by a textual note. Figure 8 illustrates the planning in the sequence diagram of MAS-ML 2.0.

Figure 8: Planning in the sequence diagram.

The utility function element is represented in the sequence diagram by an arrow with full head that begins in the agent and ends in itself, together with the stereotype <<utility-function>>. Figure 9 illustrates the representation of the utility function in the sequence diagram of MAS-ML 2.0.

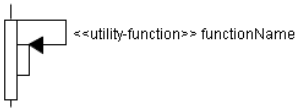

Figure 9: Utility-function in the sequence diagram.

The agent actions in MAS-ML 2.0 are modelled using the iteration element and combined fragment, existing in UML. This representation allows the modelling of any combination of actions. An example is shown in Figure 10. Since the sequence of actions for the agents with planning is generated at runtime, the modelling of this sequence is not required.

The agent goal-based with plan maintains the representation proposed by Silva (2004), as well as the plan defined during the design phase.

In the case of agent goal-based with planning, initially runs perception, next function, formulate goal

function, formulate problem function and then execution of its planning, which results in the execution of possible actions associated with the agent.

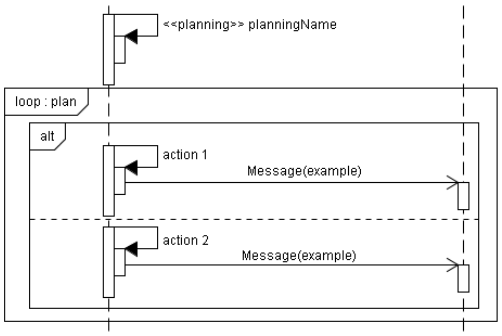

Figure 10: Implementation of the actions of the agent with planning the sequence diagram of MAS-ML 2.0.

Finally, the agent-based utility needs the perception, next function, formulate goal function, formulate problem function, planning, utility function and results in actions that are performed in that order.

4.4 Dynamic Representation of AgentClass

The features proposed by Silva, Choren and Lucena (2005) for the activity diagram, were maintained. Thus, each activity is represented by a rounded rectangle. The agent beliefs are represented by a square with the identification of the agent used by the beliefs and goals in the upper right corner through a textual description denoted with the <<Goal>> stereotype.

4.4.1 Reflex Agent Representation

The activity diagram of simple and model-based reflex agent represents the behavior from perception to action. The behavior of a simple reflex agent is represented on the activity of MAS-ML 2.0 as follows: the initial activity is the perception of the agent, on the basis of the current perception the condition action rules are used to select one of the possible actions. Finally, the selected action is performed.

In another hand, the behavior of a model-based reflex agent is represented on the activity of MAS-ML 2.0 as follows: the initial activity is the perception of the agent that can be used by the next function to update its beliefs. After that, the condition-action rules are responsible to select one of the possible actions. Finally the selected action is performed.

4.4.2 Proactive Agent Representation

The activity diagram of the goal-based agent with planning represents the agent behaviour from perception to action. The behavior of a goal-based agent is represented on the activity of MAS-ML 2.0 as follows: the initial activity is the perception of the agent, after that the next function updates the beliefs based on current perception. The formulate goal and the formulate problem functions are executed. The planning is performed to determine the action(s) should be taken. Finally, the selected action(s) is performed.

The behaviour of utility-based agent is represented on the activity of MAS-ML 2.0 as follows: the initial activity is the perception, then, the next function updates beliefs based on current perception. The formulate goal function and the formulate problem function are executed. The planning is performed to determine what action should be taken. The utility function helps the choice of action, and the selected actions are performed.

5 CASE STUDY

A TAC-SCM application is used to illustrate the use of MAS-ML 2.0 where agents with different architectures are elicited to model different strategies in the problem.

5.1 TAC-SCM

TAC (Trading Agent Competition) (Wellman et al., 2002) is an environment that enables the achievement of simultaneous auctions, test techniques, algorithms and heuristics to use in negotiation. There are two types of games in competition: TAC-Classic (Wellman et al., 2002) and TAC-SCM (Sadeh et al., 2003).

The TAC-SCM is concerned in planning and managing the organization activities across a supply chain. The TAC-SCM scenario is designed to capture the challenges in an integrated environment for acquisition of raw materials, production and delivery of finished goods to customers. This environment is highly dynamic, stochastic and strategic (Arunachalam, 2004).

The game starts when one or more agents connect to a server game. The server simulates suppliers and customers, providing a bank, manufacturing and service of storage of goods to individual agents. The game occurs along on a fixed number of simulated days, and in the end, the agent with largest sum of money in the bank is the winner (Collins et al., 2006).

5.2 Modelling TAC-SCM with MAS-ML

The internal architecture of each agent in TAC-SCM is elected according to the function in the game.

The DeliveryAgent needs to satisfy the goal of delivery products to customers. In order to achieve this

goal a sequence of actions must be executed. Thus the representation of the DeliveryAgent is modelled by using a goal-based with plan architecture (Figure 11).

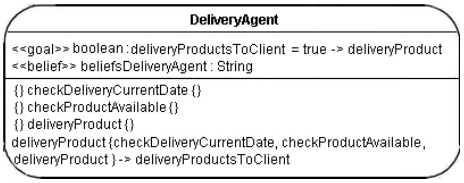

Figure 11: A DeliveryAgent proposed to TAC-SCM.

The SellerAgent offers computers to customers and gets the payment, and the BuyerAgent decides when to make new request for quote and realizes the payment. Since, reactive agents reply quickly to the perceptions (Weiss, 1999), the BuyerAgent and SellerAgent are modelled as reactive agents aiming the necessity of a fast reply in auction. Figure 12 and 13 shows the BuyerAgent (Model-based reflex agent) and SellerAgent (Simple reflex agent), respectively.

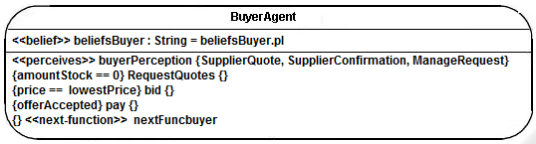

Figure 12: A BuyerAgent proposed to TAC-SCM.

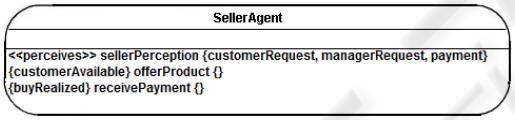

Figure 13: A SellerAgent proposed to TAC-SCM.

The ProductionAgent needs to satisfy current demand across the assembling of computers and management of the stock. To objectify achieve this goal, it can't use a pre-established plan because this dynamic scenario requires a different set of actions depending on the current demand. Thus the ProductionAgent is a goal-based with planning agent detailed in Figure 14.

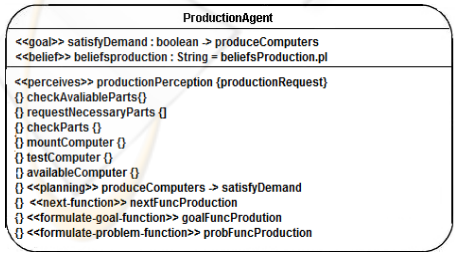

Figure 14: A ProductionAgent proposed to TAC-SCM.

Finally, the ManagerAgent is incumbed for manage all agents and the allocation resources. This agent tries to maximize gain and sales. Note that its goals can be in conflict. Thus, the most appropriate architecture in this case is the Utility-based architecture (Figure 15).

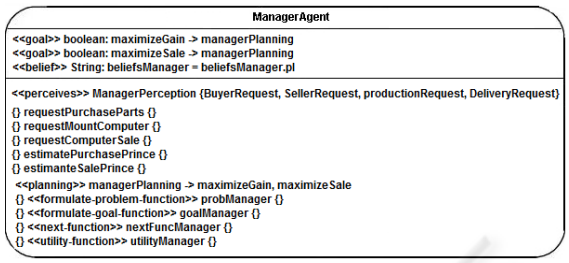

Figure 15: A ManagerAgent proposed to TAC-SCM.

The roles of reactive agents: SellerAgent and BuyerAgent are illustrated in the figures 16 and 17, respectively. The roles for proative agents are not represented since they are not affected.

Figure 16: Role of SellerAgent proposed to TAC-SCM.

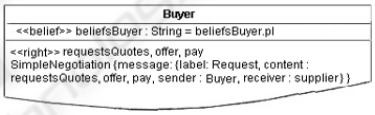

Figure 17: Role of BuyerAgent proposed to TAC-SCM.

Finally, Figure 18 depicts the Organization Diagram for TAC-SCM MAS. This diagram represents the TacOrganization and describes the agents and agent roles in the specific environment.

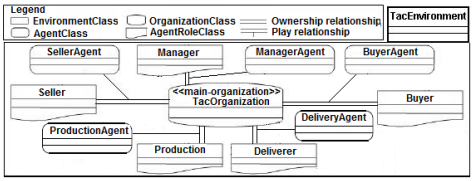

Figure 18: Organization Diagram proposed to TAC-SCM.

The activity diagrams in figures 19, 20, 21, 22 and 23 describe the behavior of each agent role.

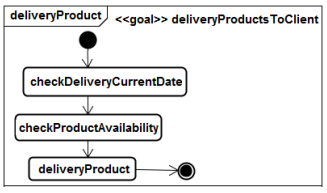

Figure 19: Role of DeliveryAgent proposed.

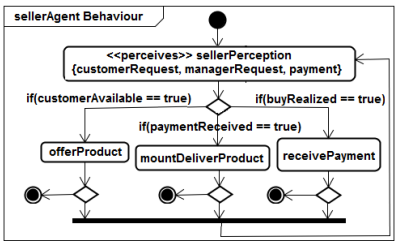

Figure 20: Role of SellerAgent proposed to TAC-SCM.

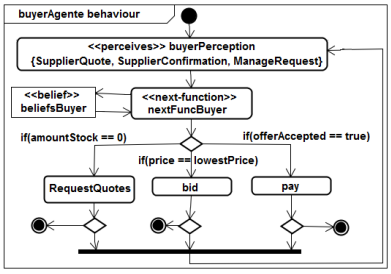

Figure 21: Role of BuyerAgent proposed.

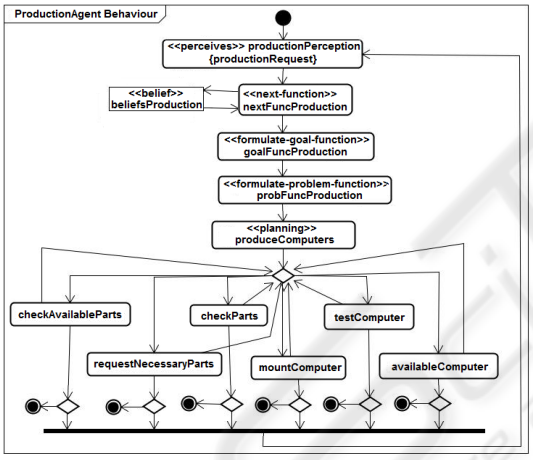

Figure 22: Role of ProductionAgent proposed.

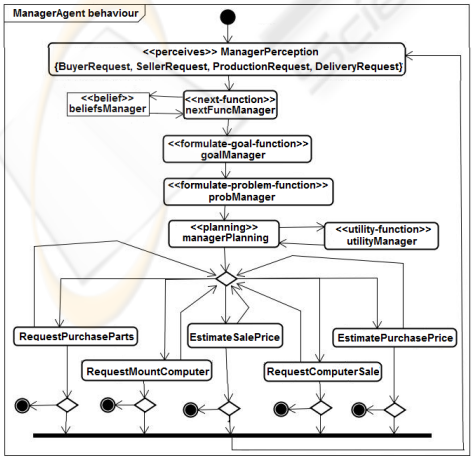

Figure 23: Role of ManagerAgent proposed.

Figure 24 describes the sequence diagram of DeliveryAgent. Note that the actions taken by the agent are guided by a plan, so it is a sequence of actions.

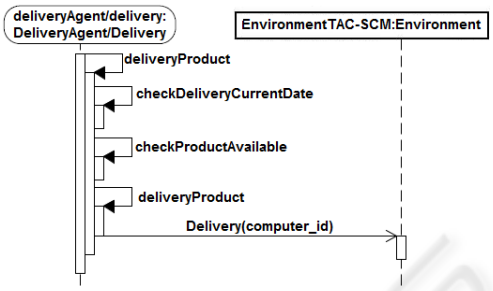

Figure 24: Sequence Diagram of DeliveryAgent.

The sequence diagram of the SellerAgent (Figure 25) shows their execution through its perceptions and a set of actions associated with a condition-action rule.

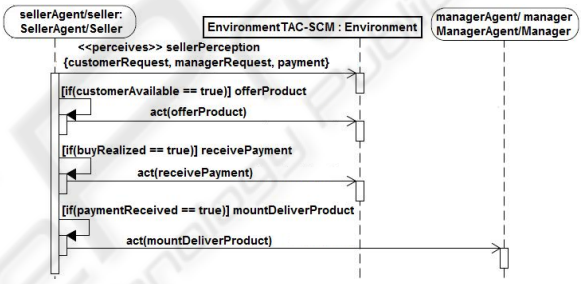

Figure 25: Sequence Diagram of SellerAgent.

The sequence diagram of the BuyerAgent (Figure 26) illustrates the agent perception, next function and a set of actions associated with a condition-action rule.

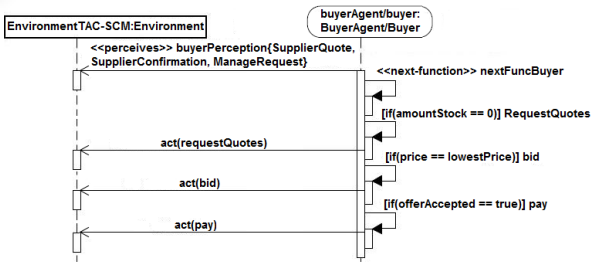

Figure 26: Sequence Diagram of BuyerAgent.

Figure 27 shows the sequence diagram of the ProductionAgent. Note that the actions taken by the agent are result of the perception, next function, formulate goal function, formulate problem function and planning. Moreover, its actions are represented by a set of possible actions.

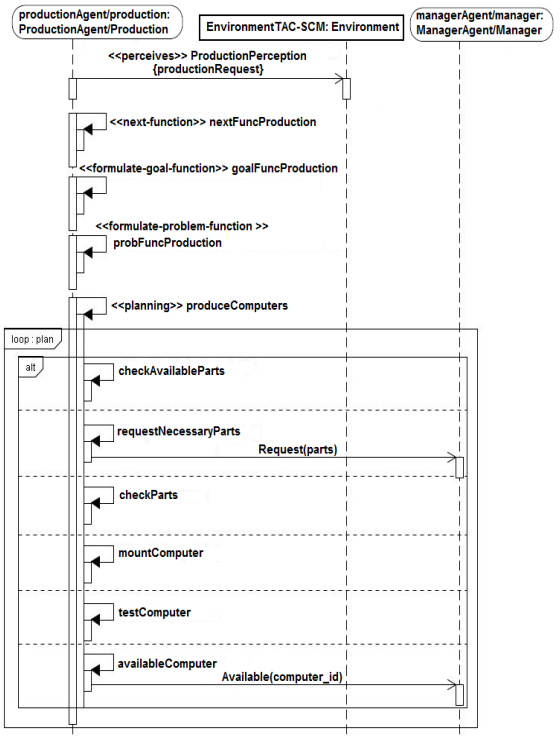

Figure 27: Sequence Diagram of ProductionAgent.

In Figure 28 is presented the sequence diagram of ManagerAgent. In this case, the actions of the ManagerAgent are guide by perception, next function, formulate goal function, formulate problem function, planning and utility function. Moreover, its actions are represented by a set of possible actions.

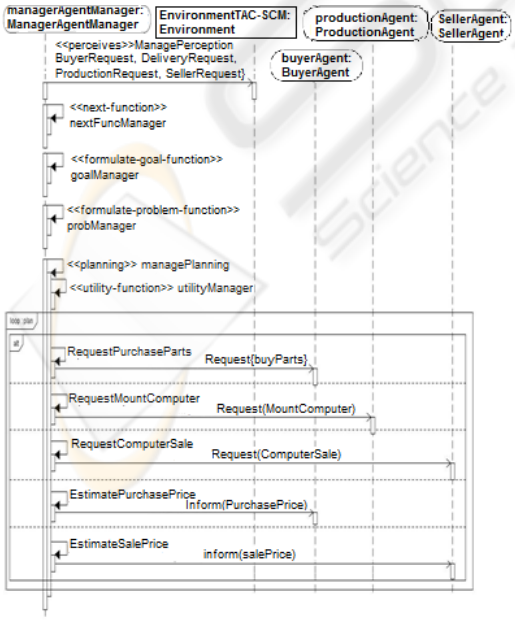

Figure 28: Sequence Diagram of ManagerAgent.

6 RELATED WORKS

Several languages have been proposed for the modelling of MAS. However, it does not support the modelling of different internal architectures of agents available in Russell and Norvig (2003) and Weiss (1999). Besides, they have several drawbacks that have justified the choosing of MAS-ML to be extended in order to model different agent architectures.

The work of Odell, Parunak and Bauer (2000) presents the AUML language. This modelling language aims to provide a semi-formal and intuitive semantics through a friendly graphical notation. AUML does not provide elements to represent perceptions and the next-function.

Wagner (2003) proposes the AORML modelling language, which is based on the AOR metamodel. This language does not give support to modelling of the elements of the internal agent architectures. Therefore it is not possible to differentiate agents with reactive and proactive architectures in AORML.

Moreover, the two languages mentioned above do not define the environment as an abstraction, so it is not possible to model the agent migration from an environment to another. This capability is inherent to mobile agents modelling (Silva. and Mendes, 2003).

Choren and Lucena (2004) present the ANote modelling language, involving a set of models, called views. In ANote, it is not possible to differentiate agents with reactive architectures from the proactive ones. In addition, ANote does not support conventional objects, used to model non-autonomous entities. The language defines several concepts related to agents, but the concept of agent role is not specified. This concept is extremely important when modelling societies where agents can play different roles at the same time.

AML (Cervenka et al., 2004) is a modelling language based on a metamodel that enables the modelling of organizational units, social relations, roles and role properties. AML gives adequate support for the modelling of reactive agents, goal-based agents with planning and utility-agents. It is worth mentioning though the semantic aspects of communication are modelled as specializations of existing elements in UML, such as methods invocation, what is not adequate for modelling agent communication.

7 CONCLUSIONS

This paper presents an extension to MAS-ML language in order to allow the modelling of diverse internal agent architectures published in the agent literature, such as: Simple reflex agents, Model-based reflex agents, Goal-based agents and Utility-based agents.

MAS-ML was originally designed to support the modelling of pro-active goal-based agents with plan. Thus, some issues were detected while trying to use the language to model reactive agents and other pro-active architectures. In this sense, the MAS-ML evolution proposed in this work involves the definition of two new metaclasses AgentPerceptionFunction and AgentPlanningStrategy in order to aggregate the representation of different agent behaviour. Also, new stereotypes to describe the behaviour of agent from specific architectures were defined and associated to AgentAction metaclass. The static structure of AgentClass and AgentRoleClass entities were also modified. Then, the class, organization, role, sequence and activity diagrams were changed in consistency.

The modelling tool to support the proposed approach is also under development. Other case studies are being conducted to provide further validation to this work. Moreover, the possibility of MAS-ML extension for other internal architectures, such as the BDI architecture is an interesting possibility.

REFERENCES

Arunachalam, R, 2004. The 2003 supply chain management trading agent competition. In: *Third International Joint Conference on Autonomous Agents & Multi Agent Systems*. July 2004. [S.l.: s.n.]. p. 113–120.

Cervenka, R., Trencansky, I., Calisti, M., and Greenwood, D 2004. AML: Agent Modeling Language Toward Industry-Grade Agent-Based Modeling. In: *Agent-Oriented Software Engineering, pp. 31–46. Springer-Verlag*, Berlin.

Choren, R. and Lucena, C., 2004. "Agent-Oriented Modeling Using ANote", 3rd International Workshop on Software Engineering for Large-Scale Multi-Agent Systems. *The Institution of Electrical Engineers, IEE, Stevenage*, UK, 2004, pp. 74-80.

Collins, J.; Arunachalam, R.; Sadeh, N.; Eriksson, J.; Finne, N.; Janson, S., 2006. The Supply Chain Management Game for the 2007 Trading Agent Competition. Available in *http://www.sics.se/tac/tac07scmspec.pdf.*

Dorigo, M. and Stützle, T., 2004. Ant Colony Optimization. *The MIT Press*, Cambridge, Massachusetts.

Jennings, Nicholas R., 1996. Coordination Techniques for Distributed Artificial Intelligence. In: *Foundations of Distributed Artificial Intelligence*, pp. 187-210, Wiley.

Odell, J., Parunak, H. V. D., Bauer, B., 2000. Extending UML for Agents. In *Agent-Oriented Information Systems Workshop at the 17th National conference on Artificial Intelligence (AIII'00)* (3-17).

Russell, S. and Norvig, P., 2003. Artificial Intelligence: A Modern Approach, *2nd Ed., Upper Saddle River, NJ: Prentice Hall*, ISBN 0-13-790395-2,

Sadeh, N.; Arunachalam, R.; Erikson, J.; Finne, N.; Janson, S., 2003. A supply-chain trading competition. *AI Magazine*, v. 24, n. 1, p. 92–94

Silva, P. S. e Mendes, M. J. (2003). Uma Abordagem para Incorporar Mecanismos de Inteligência Artificial a Agentes Móveis. *XXI Simpósio Brasileiro de Redes de Computadores*. Natal, Rio Grande do Norte.pp 837-852.

Silva, V.; Lucena, C. 2004. From a Conceptual Framework for Agents and Objects to a Multi-Agent System Modeling Language, In *Journal of Autonomous Agents and Multi-Agent Systems*, volume 9, issue 1-2, Kluwer Academic Publishers, pp. 145-189, 2004.

Silva, V.; Choren, R.; Lucena, C. 2004. A UML Based Approach for Modeling and Implementing Multi-Agent Systems. In: *Proceeding of the third International Conference on Autonomous Agents and Multi-Agents Systems*. New York, USA, IEEE Computer Society, volume 2, pp. 914-921.

Silva, V. T. da, Choren, R., Lucena, C. J. P. de (2005). Using UML 2.0 Activity Diagram to Model Agent Plans and Actions. In: *4th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, Netherlands, pp. 594-600.

Silva, V.; Choren R.; Lucena, C. 2008 a. MAS-ML: A Multi-Agent System Modelling Language, In *International Journal of Agent-Oriented Software Engineering, Interscience Publishers*, vol.2, no.4.

Silva, V.; Choren, R.; Lucena, C. 2008 b. Modeling MAS Properties with MAS-ML Dynamic Diagrams. In *8th International Bi-Conference Workshop, LNCS 4898, Springer-Verlag*, pp. 1-18.

Wagner, G., 2003. The Agent-Object-Relationship Meta-Model: Towards a Unified View of State and Behavior. Information Systems, v. 28, n.5, pp. 475–504.

Weiss, G., 1999. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. *MIT Press, Massachusetts.*

Wellman, M. P.; Stone, P.; Greenwald, A.; Wurman, P. R., 2002. The 2001 Trading Agent Competition. *IEEE Internet Computing*, v. 13, p. 935–941.

UML: Unified Modeling Language Specification, version 2.2, OMG, 2009 available in: *http:// www.omg.org/technology/documents/modeling_spec_catalog.htm#UML.*

Zambonelli, F.; Jennings, N.; Wooldridge, M., 2001. Organizational abstractions for the analysis and design of multi-agent systems. In: *Agent-Oriented Software Engineering, LNCS 1957, Berlin: Springer*, p. 127-141.