# Extending UML to Model Multi-Agent Systems

Viviane Torres da Silva, Carlos J. P. de Lucena

Computer Science Department – Pontifical Catholic University of Rio de Janeiro (PUC-Rio)
Rua Marquês de São Vicente, 225 – Gávea, Rio de Janeiro / RJ, 22451-900, Brazil
{viviane,lucena}@inf.puc-rio.br
PUC-Rio Inf.MCC 08/04 March, 2004

**Abstract.** Multi-agent systems (MASs) and object-oriented systems (OOSs) differ in many respects. Traditionally, OOSs are composed of objects whose properties are attributes and methods and interact through method calling. MASs are composed not only of objects but of different elements (such as agents, organizations and others) that have different properties and interact in different ways. MASs require modeling languages, among other agent-based software technologies, that are able to explore the use of agent-related abstractions. Modeling languages should represent the structural (or static) and dynamic aspects of MASs by expressing the characteristics of all its essential entities. Structural aspects comprise the definition of the entities, their properties and their relationships. The dynamic aspects are characterized by the internal execution of the entities and by the interactions between the entities. In this paper we propose to extend the UML modeling language in order to model the structural and dynamic aspects of MASs.

**Keywords.** Modeling language, multi-agent system, object-oriented systems, UML, meta-model

**Resumo.** Sistemas multi-agentes (SMAs) e sistemas orientados a objetos (SOOs) diferentes em muitos ...Tradicionalmente, SOOs são compostos por objetos cujas propriedades são atributos e métodos e interagem através da chamada de métodos. SMAs são compostos não apenas por objetos mas também por diferentes elementos (como agentes, organizações e outros) que possuem propriedades diferentes e interagem de maneiras diferentes. SMAs requerem linguagem de modelagens, entre outras técnicas de software baseadas em agentes, que sejam capazes de explorar o uso de abstrações relacionadas a agentes. Linguagens de modelagens devem representar os aspectos estruturais (ou estáticos) e dinâmicos de SMAs expressando as características de todas as entidades essenciais. Os aspectos estruturais compreendem a definição das entidades, das propriedades e dos relacionamentos. Os aspectos dinâmicos são caracterizados pela execução interna das entidades e pela interação entre elas. Neste artigo nos propomos estender a linguagem de modelagem UML para modelar os aspectos estáticos e dinâmicos de SMAs.

**Palavras-chave.** Linguagem de modelagem, sistemas multi-agentes, sistemas orientados a objetos, UML, meta-modelo.

## 1. Introduction

Multi-agent systems (MASs) and object-oriented systems (OOSs) differ in many respects. When comparing OOSs to MASs, we note that MASs are composed of different elements that have different properties and characteristics and are related though different relationships. The heterogeneity of the elements' properties and characteristics make MAS systems fundamentally different from the OOSs. MASs are composed not only of objects but of agents [13,19], organizations [1,12,21], agent roles [12,20,21], object roles [7,9,10] and environments [12,20]. Agents are autonomous, interactive and adaptive entities [13] that play roles in organizations [6,7,13] in order to try to achieve their goals. Organizations group together the agents of a MAS [1,12]. Organizations are entities that define the roles that can be played by agents and sub-organizations [4] and describe some axioms (laws or rules) [11,21] that agents and sub-organizations must obey. Environments are the habitat of agents, organizations and objects [3].

Since MASs differ from OOSs, the viability of MAS's application adoption depends on the development of new techniques, methodologies, modeling languages, development platforms, tools and programming languages to support the specification, analysis, design and implementation of software agents and the systems in which they are embedded. In particular, MASs modeling languages need to explore the use of agent-related abstractions. Such modeling languages should represent the structural (or static) and dynamic aspects of MASs by expressing the characteristics of all its essential entities. Structural aspects comprise the definition of the entities, their properties and their relationships. The dynamic aspects are characterized by the internal execution of the entities and by the interactions between the entities.

In this paper we propose a MASs modeling language called MAS-ML that extends the UML modeling language [17]. The use of UML as a basis for the creation of MAS-ML reduces the common risks of adopting new technologies. MAS-ML is an extension of a well-know and *de facto* standard for object-oriented modeling. UML is used both in industry and academy for modeling object-oriented systems.

UML can be used when creating a MASs modeling language because objects and agents co-exist in MASs. However, UML does not provide sufficient support for modeling MASs. Among other things, UML does not provide support for modeling agents, their properties, the relationships between agents and other MASs and the interactions between them.

MAS-ML extends UML according to the TAO (Taming Agents and Objects) conceptual framework [15]. The goal of the conceptual framework is to define a core set of MAS abstractions. The core set of abstractions used in TAO has been developed based upon our investigation of existing agent-based and object-oriented methodologies [1,20], languages [8][14], and theories [2].

To produce a conservative extension of UML [16], we focus on the adaptation of the static class diagram and the interaction sequence diagram according to TAO concepts. These two diagrams have been chosen because they are the most commonly used and because it is possible to use them to illustrate both the structural and dynamic aspects of the TAO meta-model (or conceptual framework).

Several modeling languages for MAS that extend the UML meta-model such as [12,18] have been proposed. However, there is still a need for a modeling language that (i) describes agent-related concepts as first-class abstractions, (ii) is based on an explicit description of a MAS meta-model, and (iii) models the structural and dynamic aspects frequently described in MASs.

The paper is organized as follows. Section 2 presents the MASs entities, their properties and their relationships and defines the extensions applied to the UML meta-model. Section 3 and 4 describe the extensions of the UML class diagram and of the UML sequence diagram, respectively. Section 5 reviews some related work. Finally, Section 6 discusses some future directions and presents the conclusions of our work.

## 2. The UML Meta-model Extension

The UML meta-model [17] was extended to incorporate the MAS entities defined in TAO, their properties and their relationships. The entities defined by the TAO conceptual framework are object, agent, organization, agent role, object role and environment. Besides, TAO describes a set of eight relationships used to link these entities: inhabit, ownership, play, control, dependency, association, aggregation and specialization relationships. For further details about the description of the entities and relationships see [15].

The definition presented in TAO for the entity *object class* is similar to the definition presented in the UML meta-model for the meta-class called *Class*. So, a direct mapping was defined from this TAO entity to the respective UML meta-class. In addition, the association, aggregation, dependency and specialization relationships described in TAO are also defined in UML and therefore the UML meta-model was not extended to incorporate these relationships.

All agent-related abstractions defined in MAS-ML meta-model extend the meta-class *Classifier*. The meta-class *Classifier* is related to the meta-classes *StructuralFeature* and *BehavioralFeature*. A structural feature is a typed feature of a classifier that specifies the structure of instances of the classifier. A behavioral feature is a feature of a classifier that specifies an aspect of the behavior of its instances. The *StructuralFeatures* meta-class is a generalization of the *Property* meta-class (*attributes* of a class are represented as instances of *Property*) and the *BehavioralFeatures* meta-class is a generalization of the *Operation* meta-classes, according to the UML definition [17].

In the following sub-sections the extensions proposed to the UML meta-model are presented. The meta-classes and stereotypes created to represent the entities, properties and relationships defined in TAO are described (Figure 1 and 2). In addition, associations between the new meta-classes are also defined (Figure 3).

### 2.1. Agent

Based on the difference between *agents* and *objects*, we did not create a stereotype `<<agent>>` and refer it to the UML meta-class *Class* as others proposals such as AUML [5,12] and AOR [18] have done. If stereotype was used, agents would have the same properties and relationships defined on the meta-class *Class* and also additionally values and constraints defined by the stereotypes. Moreover, we have not extended the meta-class *Class* since class is a kind of *Classifier* whose features are attributes and operations [17]. Agent and objects define different structural and behavioral features.

In order to extend UML meta-model to describe agents, a new meta-class called *AgentClass* was created (Figure 1). The *AgentClass* extends the UML meta-class *Classifier* and therefore, is associated with the meta-classes *StructuralFeature* and *BehavioralFeature*. The structural features defined by an agent are *goal* and *belief* and its behavioral features are *action* and *plan*. *Goals and beliefs* are features of agents, organizations, active environments and agent roles. However, the feature goal related to agents, organizations and active environments has not the same definition of the feature goal related to agent roles. A goal associated with an agent, organization or active environment is related to plans that achieve the goal but a goal associated with an agent role is not related to plans.

In order to represent goals and beliefs, a stereotype called `<<goal>>` and a stereotype called `<<belief>>` were defined based on the meta-class *Property*. A property owned by a class is an attribute [17]. The stereotype `<<belief>>` is a simple extension of *Property*. It does not specify any tag value or constraints. It only identifies the attributes that are *beliefs*. The stereotype `<<goal>>` describes one tag and one constraint. The tag *planTag* links the goal to a set of *plans* that achieves the *goal*. The constraint states that the stereotype `<<goal>>` is defined by an agent, an organization, an active environment or by an agent role. When an agent, organization or active environment defines a goal, the tag *planTag* is associated with a set of plans, but when an agent role defines a goal, the *planTag* must be empty since roles do not define plans.

*Actions* are behavioral features of agents. However, *actions* cannot be defined as a stereotype based on the meta-class *Operation* since the definition of *actions* and *operations* are different. An operation may be implemented by a method that can be invoked/requested by an object. Actions associated with agents are never called by another agent, but rather only executed under the control of the agent itself. Agents interact by sending and receiving messages and not by calling the execution of actions. Moreover, we have not used the meta-class *Action* defined in the UML to represent actions of agents because the meta-class does not extend the meta-class *BehavioralFeature* and therefore cannot be described as a feature of a *Classifier*.

We have created a new meta-class called *AgentAction* that extends the meta-class *BehavioralFeature* to represent the actions executed by agents. Similar to the meta-class *Operation*, the new meta-class *AgentAction* is associated with the meta-class *Constraints* in order to define preconditions and post-conditions. A constraint is a condition or restriction expressed in natural language text or in a machine readable language for the purpose of declaring some of the semantics of an entity [17].

The behavioral features of agents are also composed of their *plans*. A new meta-class called *AgentPlan* has been created to specify the *agent* feature plans in UML. A *plan* is associated with *goal* and is represented by a sequence of *actions* that is executed by an *agent* to achieve the *goal*. A new meta-class to represent plan was created because there is no meta-class in the UML meta-model with the same meaning of *plan*. The meta-class *AgentPlan* is a specialization of the *BehavioralFeature* meta-class. A stereotype to describe *plan* based on the definition of the meta-class *AgentAction* was not created because a *plan* and an *action* do not share the same properties and relationships.

Every agent plays at least one role in an organization. An agent can play more than one role in different organizations but an agent role can not be played by more than one agent. In order to represent agents playing roles in organizations the *AgentClass* meta-class is associated with the meta-classes that describe agent roles and organizations. In addition, the *AgentClass* meta-class is also associated with the meta-class that represents environments because every agent must inhabit exactly one environment (Figure 2).

### 2.2. Object Role

An *object role* guides and restricts the behavior of an object since it manipulates the object and describes a set of features that are viewed by other entities. An object role can restrict the features of an object but can also add information (attributes) and behavior (methods) to the object that plays the role.

In order to comply with these ideas, we have two options: to create a new meta-class or to adapt the meta-class *ConnectableEntity* defined in UML. The meta-class *ConnectableEntity* should be adapted because it does not describe the properties (or features) viewed by other objects. The meta-class does not describe the object properties restricted by the object role or the properties added to the set of object properties.

We propose to create a new meta-class called *ObjectRoleClass* because we are interested in describing a conservative extension [16] to the UML meta-model. The new meta-class *ObjectRoleClass* is a specialization of *Classifier* (Figure 1). So, the *ObjectRoleClass* is associated with the meta-classes *BehavioralFeature* and *StructuralFeature*. The features of *ObjectRoleClass* represent the view other entities have of the object playing the role. The *ObjectRoleClass* meta-class can restrict the access to attributes (*StructuralFetures*) and methods (*BehavioralFeatures*) described in the class of the object and can also add new attributes and methods to the class.

Object roles are *defined* in the context of an organization and are played by objects. An object can *play* more than one role in different organizations but an object role can only be played by one object [15]. In order to represent the relationships between object roles, organizations and objects, two associations linking the *ObjectRoleClass* meta-class and the meta-classes that represent organizations and objects are provided (Figure 2).

### 2.3. Agent Role

An *agent role* is concerned with guiding and restricting the autonomous behavior of an *agent* (or an organization) by describing the goals of an agent (or organization), additional beliefs, the actions that an agent (or organization) must perform and the actions that an agent (or organization) can perform while playing the role.

A new meta-class called *AgentRoleClass* was created to represent the roles played by agents. The *AgentRoleClass* extends the *Classifier* and so describes structural features and behavioral features (Figure 1). The structural features of the *AgentRoleClass* meta-class are the *goals* and *beliefs* related to agent roles. The behavioral features of the *AgentRoleClass* meta-class describe the *duties*, *rights* and *protocols* related to agent roles. *Duty* and *right* are defined as stereotypes of the meta-class *AgentAction*. The `<<duty>>` stereotype identifies *actions* that *agents* playing the *role* must perform, and the `<<right>>` stereotype identifies *actions* that *agents* playing the *role* have permission to perform.

A new meta-classs called *AgentProtocol* has been created to describe the *protocols* that the *agent* must obey when playing the *role* and interacting with other system entities. The meta-class *AgentProtocol* specializes the meta-class *BehaviorFeature* because it is part of the behavior of *agent roles*. An agent *protocol* defines the set of *messages* that an *agent* is allowed to send to another *agent* in an interaction and the *messages* that it can receive from another *agents*.

The *messages* sent and received by an *agent* are different from the *messages* sent and received by an *object*. The UML meta-model defines a message sent and received by objects as a specific kind of communication in an interaction. A communication can be raising a signal, invoking an *Operation* or creating and destroying an *instance* [17]. Thus, a *message* sent and received by an *object* may be related to an *operation* and consequently also related to a *method* executed by the *object* that receives the *message*. On the other hand, a *message* sent and received by an *agent* is never connected to the *actions* executed by the agent that receives the *message*.

In order to make the difference between the *messages* of *objects* defined by the UML meta-model and the *messages* of *agents* explicit, we have defined a new meta-class called *AgentMessage* to represent the *messages* specified in protocols. The features related to a *message* are the *label* specifying the message kind, the content kind that can be empty, the *sender* and the *receiver* of the message [1].

Every agent role is *defined* in the context of an organization and is *played* by one agent or one sub-organization [4]. From the perspective of entities outside an organization, sub-organizations can be viewed as agents that *play* roles in the organization where they are defined. In order to represent such relationships, the *AgentRoleClass* meta-class is associated with the meta-classes that represent agents and organizations (Figure 2).

### 2.4. Organization

Although we have defined in TAO that an *organization* extends the properties and relationships defined by *agents*, we have not created a stereotype `<<organization>>` associated with the meta-class *AgentClass*. Organizations extend the notion of agents describing additional properties and relationships. An *organization* defines a set of *axioms* (*rules* and *laws*) that agents and sub-organizations must obey. The axioms characterize the global constraints of the organization. In addition, an organization also defines roles that must be played by the agents and sub-organizations within it. To represent organization, a new meta-class called *OrganizationClass* was created (Figure 1). This meta-class specifies the meta-class *AgentClass*.

Since organizations are extensions of agents, organizations have *goals*, *beliefs*, *actions* and *plans* and play *roles* in the organizations where they are defined. The organization *goals*, *beliefs* and *axioms* constitute the organization structural features. In turn, the *actions* and *plans* of the organization are its behavioral features.

To represent the *axioms* of an organization we create the stereotype `<<axiom>>` related to the meta-class *Property*. Such as goals and beliefs, axioms can also be expressed as attributes. An

axiom is identified by its name and has a value that describes itself. With the aim to interpret the value of an axiom it is important to associate a type with it.

Every organization *defines* agent roles and object roles [4]. In order to represent the domain-independent relationships between organizations and the roles that they define, associations linking the *OrganizationClass* meta-class to the *AgentRoleClass* and *ObjectRoleClass* meta-classes were defined.

Any organization can *define* several sub-organizations and all sub-organizations must be part of one organization [4]. A sub-organization is an organization that *plays* a role *in* another organization. The relationship between an organization and its sub-organizations is represented by an aggregation relationship linking the *OrganizationClass* meta-class to itself. The relationship between a sub-organization and an agent role is represented by an association relationship linking the *OrganizationClass* meta-class and the *AgentRoleClass* meta-class.

The roles *defined* by an organization are played by agents, sub-organizations and objects in the context of the organizations [4]. The relationships between organizations and the agents and objects that *play* roles *in* the organizations are represented by associating the *OrganizationClass* meta-class with the *AgentClass*, and by associating the *OrganizationClass* meta-class with the *Class* meta-class. Moreover, every organization inhabits one environment [3]. The relationship between organizations and environments is represented by associating the *OrganizationClass* meta-class and the *EnvironmentClass* meta-class (Figure 2).

### 2.5. Environment

An environment can be an active entity such as an agent or a passive entity such as a traditional object. An abstract meta-class called *EnvironmentClass* was created by extending the *Classifier* meta-class to represent environments. The *EnvironemntClass* meta-class does not define any property since they depend on the environment characteristics. However, the meta-class defines the associations between it and other MAS-ML meta-classes.

Although passive environments and objects share similar properties, a passive environment cannot be defined by creating a stereotype based on the meta-class *Class* because passive environments and objects do not define the same relationships. Objects can play roles in organizations and environments are the habitat of objects and organizations. A meta-class called *PassiveEnvironmentClass* was created to represent passive environments. The *PassiveEnvironmentClass* meta-class extends the *EnvironmentClass* meta-class. The structural features of passive environments are *attributes* and their behavioral features are *methods*.

Active environments and agents share similar properties. However, active environments and agents do not define the same relationships and, therefore, active environment could not be defined as a stereotype based on the meta-class *AgentClass*. Agents play roles in organizations and environments are the habitat of agents and organizations. The meta-class *ActiveEnvironmentClass* was created by extending the *EnvironmentClass* meta-class to represent active environments. The structural features of active environments are *goals* and *beliefs* and their behavioral features are *actions* and *plans* (Figure 1).

Every environment is the habitat of objects, agents and organizations [3]. Every agent, organization and object *inhabits* exactly one environment. Such relationships are domain-independent and, for this reason, must be represented in the MAS-ML meta-model. Associations were described to link the abstract meta-class *EnvironmentClass* to the meta-classes that define agents, organizations and objects (Figure 2).

Figure 1 presents a sub-set of meta-classes of the UML meta-model and the extensions made by MAS-ML. This figure shows the new meta-classes and the new stereotypes that have been proposed by the MAS-ML related to the entities and properties described in TAO. The icons that represent the stereotypes are associated with the meta-classes on which the stereotypes are based.

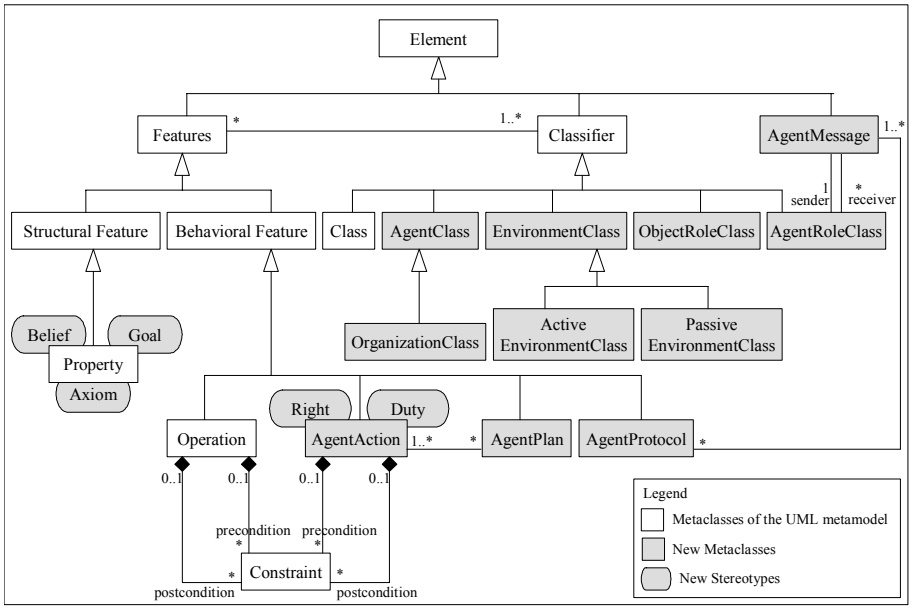


**Fig. 1.** The extended UML meta-model incorporating the MAS-ML entities and properties

Figure 2 illustrates the associations between the meta-classes that represent the TAO entities. As stated before, such associations were defined based on the domain-independent relationships between the entities defined in TAO.

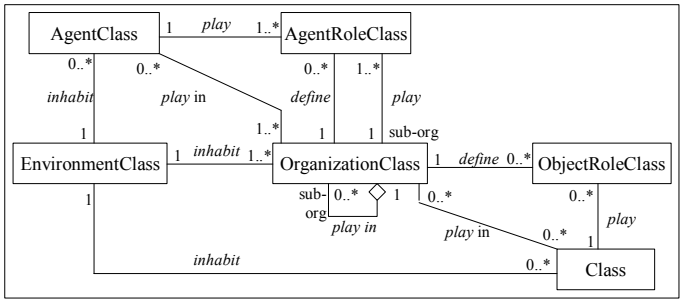


**Fig. 2.** The associations between the meta-classes that represent the TAO entities

## 2.6. The Inhabit Relationship

A new meta-class called *Inhabit* was created to represent the inhabit relationship defined in TAO. The meta-class *Inhabit* extends the meta-class *DirectedRelationship*. The inhabit relationship relates the habitat and the entities that inhabit (the citizens of) the habitat. The inhabit relationship can be applied to environment classes and agent classes, to environment classes and object classes and to environment classes and organization classes. When an environment class is related to an entity class (agent, object or organization class) by the inhabit relationship it means that all the entity instances are inhabiting environment instances of that class.

## 2.7. The Ownership Relationship

The new meta-class *Ownership* that extends the meta-class *DirectedRelationship* was created to represent the ownership relationship. The ownership specifies that an entity (the member) is defined in the scope of another entity (the owner) and that a member must obey a set of global constraints defined by its owner. An ownership relationship can be applied to organization classes and agent role classes and to organization classes and object role classes where organizations are

the owners and roles are the members. When a role class is related to an organization class by the ownership relationship it means that role instances are owned by organization instances.

### 2.8. The Play Relationship

To represent the play relationship the meta-class *Play* was created. The meta-class *Play* extends the *DirectedRelationship*. The play relationship specifies the roles that an entity can play. A play relationship can be applied to agent classes and agent role classes, organization classes and agent role classes and object classes and object role classes. When an entity class is related to a role class by the play relationship it means that the entity instance can play role instances.

### 2.9. The Control Relationship

The control relationship is represented by the new meta-class *Control* extending the *DirectedRelationship*. The control relationship defines that the controlled entity must do anything that the controller entity requests. A control relationship is applied to agent roles. An agent role instance can be the controller of another agent role instance. Figure 3 shows the new meta-classes that represent the relationships described in TAO that have been proposed by the MAS-ML to the UML meta-model.

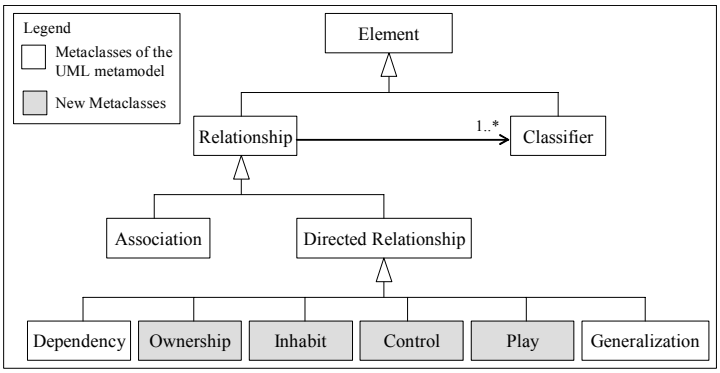


**Fig. 3.** The extended UML meta-model incorporating the MAS-ML relationships

## 3. The UML Class Diagram Extension

When introducing new abstraction in the UML meta-model, new diagram elements to represent the new entities and relationships need to be created. In principle as TAO defines a set of six entities and eight different relationships, we should need fifteen new diagram elements to represent them. Fortunately, an entity (object) and some relationships (association, aggregation, specialization and dependency relationships) defined in TAO are also presented in the UML meta-model. All diagram elements that represent MAS-ML entities were defined with three compartments: the top compartment holds the class name, the middle compartment holds the structural features and the bottom compartment holds the behavioral features of the entity.

Because of the set of different entities and relationships defined in the TAO meta-model that have been incorporated in the UML meta-model, we have proposed different structural diagrams to focus on the different extension aspects to be covered by the resulting MAS-ML. The structural diagrams that compose MAS-ML are the extended class diagram and two new diagrams called organization diagram and role diagram. The three structural diagrams – Class, Organization and Role diagrams – show all entities and all relationships defined in TAO.

Our main goal with the extension of *class diagrams* is to modify the class diagram to represent the relationships between the classes and other MAS entities. The extended class diagram represents the relationships between the classes and environments, classes and agents and classes

and organizations. Moreover, the class diagram was also extended to represent the relationships between agents, between environments and between organizations.

The classes that may participate in this diagram are agent class, organization class, environment class and other classes defined by UML. The relationships used in this diagram are those defined by UML plus the relationships inhabit (used between classes and environment classes), association (used between agent classes and classes, between organization classes and classes, and between environment classes), and specialization (used between agent classes, between organization classes and between environment classes).

The objective of the *organization diagrams* is to model all the organizations of a system. Each organization diagram is responsible for modeling one organization, i.e., to model the properties of the organization (goals, beliefs, plans, actions and axioms), the roles defined by the organization, the entities (agents, classes and sub-organizations) that play these roles, and the environment that the organization inhabit. The specialization relationships between organizations are modeled in class diagrams.

In organization diagrams it is also important to describe the properties of the roles (goals, beliefs, duties, rights and protocols) defined in the organization and the entities that play each role. The relationships between the roles and between the roles and the classes are modeled in the role diagrams. For each agent, object and sub-organization described in organization diagrams, it is necessary to define their properties. However, their relationships are modeled in class diagrams. The classes that may be used in organization diagrams are organization class, agent class, agent role class, class, object role class and environment class. The relationships that may be used are: ownership, play, and inhabit.

The *role diagram* is responsible for illustrating the relationships between the agent roles and object roles identified in the organization diagrams. This diagram also identifies the classes accessed by the agent roles and object roles. The interactions between the agents and organizations of the system are described based on the relationships between the roles illustrated in role diagrams. A role diagram can show the relationships between agent role class and object role class and between these roles and classes. The relationships used in this diagram are: control, dependency (used between object role classes, between agent role classes and object role classes and between agent role classes), association (used between object role classes, between agent role classes and object role classes, between agent role classes and between role classes and classes), aggregation (used between object role classes and between agent role classes), and specialization (used between object role classes and between agent role classes). Figure 4 illustrates an organization diagram modeling the organization *General Store*. It shows the roles that the organization defines and the entities that play the roles. Figure 5 depicts the relationships between the roles defined in the *General Store* and in the *Second-hand Bookstore* organizations. The middle and bottom compartments of the diagram elements were omitted in Figures 4 and 5.

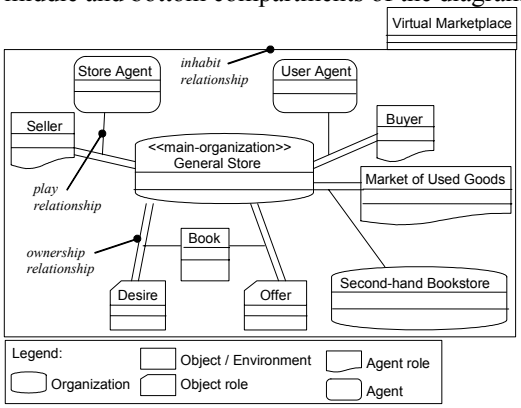


**Fig.** 4. Organization diagram

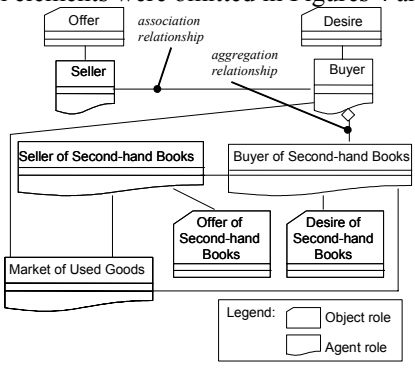


**Fig.** 5. Role diagram

## 4. The UML Sequence Diagram Extension

We propose to extend the UML sequence diagram to represent the dynamic aspects of MASs, i.e., to represent the interactions between the MAS instances and the intra-actions defined by each instance. First of all, when extending the UML sequence diagram new pathnames and icons need to be defined with the aim of representing the MAS instances (agents, organizations and environments) in sequence diagrams.

The pathname associated with an entity instance identifies it as a unique entity in the system. An instance can be identified using a complete or a simple pathname. In UML, the complete pathname completely specifies an object participating in an interaction because it describes the object by identifying the class in which it was based and its complete structure of packages. The simple pathname is a simplification version of the complete pathname where some information are omitted when the context of the interaction is well known or irrelevant.

In MAS-ML, the complete pathnames of agent, organization and environment are defined. The complete pathname of agents describe the agent instance name, the role instance that the agent is playing, the organization hierarchy where the agent is playing the role, the environment that it inhabits and their corresponding class names. The organization pathname describes the organization instance, the role that the organization is playing (if any), the complete organization hierarchy where the organization is defined, the environment that it inhabits and their corresponding class names. The pathname that completely specifies the environment is the simplest one. It describes the environment instance name and its class name.

To represent the interactions between agents, organizations, environments and objects, the definition of the concept called *message* used in the UML was extended to represent entities sending and receiving messages and not only calling methods of other entities. In the extended sequence diagram proposed by MAS-ML, an arrow can be used to represent a method of an object or passive environment being called by other entity or can be used to represent a message being sending by an agent, organization or active environment to another agent, organization or active environment. In UML, filled head arrows represent synchronous messages and open head arrow represent asynchronous messages. In MAS-ML, an open head arrows is used to illustrate an agent message by associating the label of the message and the description of the content with it. The directed line indicates the sender and the receiver of the message.

To represent the creation and destruction of MAS instances and to represent the interaction between agents, organizations, objects and their roles some stereotypes associated with messages were redefined and others were created. The stereotypes `<<create>>` and `<<destroy>>` were extended to represent the creation and destruction of MASs entities. The stereotypes `<<role_commitment>>` and `<<role_cancel>>` represent agents, object and organizations committing to a role and canceling the commitment, respectively. The stereotypes `<<role_activate>>` and `<<role_deactivate>>` represent agents and organizations activating an inactive role and deactivating an active role. The stereotype `<<role_change>>` represents agents and organizations changing their roles, i.e., canceling or deactivating a role and creating or activating a role. Using such stereotypes it is possible (i) to model an agent (or a sub-organization) entering an organization to play a role by creating or activating a role, (ii) to model an agent (or sub-organization) leaving an organization by canceling or deactivating a role, (iii) to model an agent (or sub-organization) moving from a environment to another by canceling or deactivating all its roles in the actual environment and creating or activating at least one role in the other environment.

The sequence diagram was also extended to represent the execution of plans and actions while modeling the intra-actions related to agents, organizations and environments. The execution of a plan or an action is represented by an arrow from the agent returning to itself by beginning another focus of control (or activation bar). The focus of control of a plan defines the sequence of actions that will be executed by the entity in the context of the plan. The focus of control of an action defines what the entity will do when executing the action. The calling of a plan or an action is adorned with the name of the plan or action in order to indicate what will be executed.

The UML sequence diagram was extended to illustrate the protocols described by agent roles. When using the extended diagram to model a protocol, the diagram represents the agent roles involved in the protocol and the sequence of messages defined by the protocol. Figure 6 illustrates the protocol called "to enter in market of used goods" between two roles. This protocol defines the sequence of messages sent and received by entities playing the roles *Buyer* and *MarketOfUsedGoods*. Figure 7 illustrates a user agent playing the role *Buyer* interacting with a second-hand bookstore playing the role *MarketOfUsedGoods*. The figure shows the plans and actions executed by the agent and organization and the messages sent by them according to the protocol illustrated in Figure 6. Figure 7 also depicts the agent committing to a new role and canceling a role.

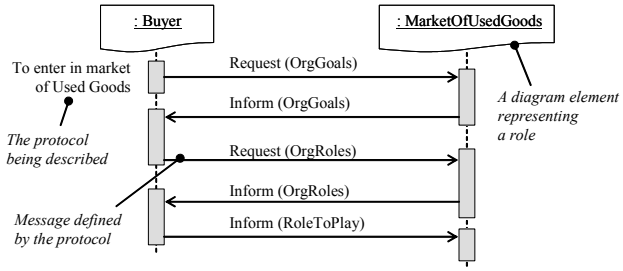


**Fig. 6.** A protocol between two roles

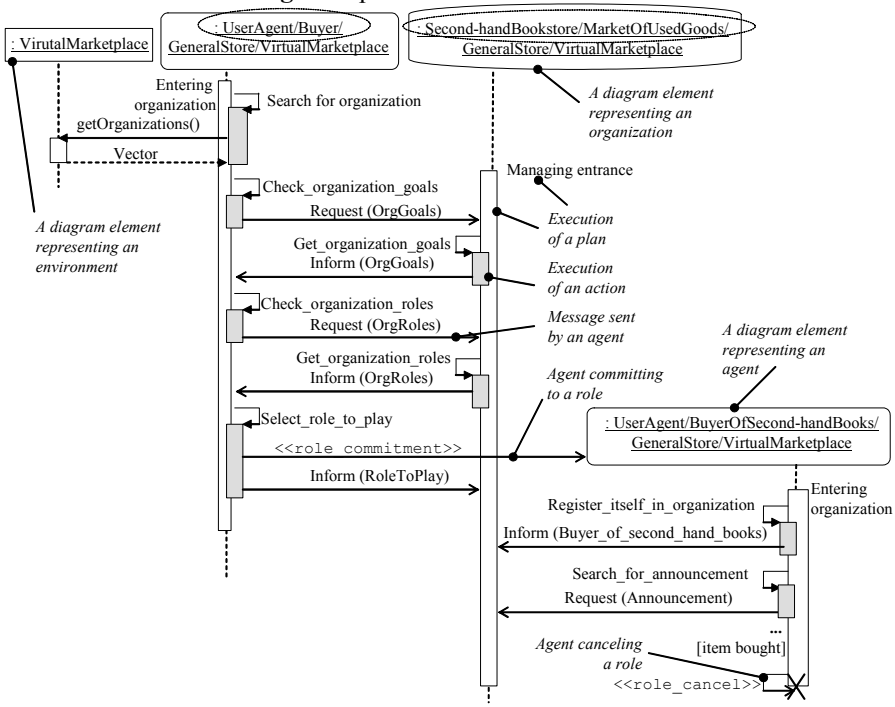


**Fig. 7.** An agent interacting with an organization, committing to a new role and canceling a role

## 5. Related Works

After analyzing many of the MAS modeling languages published in the literature, it was possible to realize that none of the analyzed modeling languages contemplates all concepts defined in TAO, i.e., they do not contemplate several concepts usually found in MASs. The modeling languages do not model the structural aspects (entities and relationships) nor the dynamic aspects (domain-independent behavior) frequently described in MASs and defined in TAO. Some proposals describe a sub-set of those abstractions and others do not model the interaction between the entities that they define. AOR [18] and AUML [12] are the better know modeling languages for MASs. AOR and AUML do not define environments as an abstraction. Therefore, it is not possible to

model an agent moving from an environment to another using the proposed sequence diagram. Using AOR models it is also not possible to model agents moving from an organization to another. In addition, AOR does not model the internal execution of agents (and other MAS entities) and the protocols defined by roles. Stereotypes based on the meta-class *Class* are used to define all the AOR entities. In Section 2.1 we justify why stereotypes were not used to represent agents.

In AUML organizations are not modeled in sequence diagram. Thus, it is not possible to model the interaction between agents and organizations. Moreover, the language does not define nor model the intra-actions (internal execution) of the entities that they define. In addition, agents are defined as a stereotype of the meta-class *Class* and roles played by agents are defined based on the meta-class *ClassifierRole*. In Section 2.3, we describe the difference between agent roles and object roles. AUML does not describe the UML extension that model organizations. Besides, the properties of roles and organizations are not defined in AUML.

## 6. Conclusion and Future Work

Using MAS-ML it is possible to represent the frequently used structural and dynamic aspects of MASs as defined in TAO. Using the three structural diagrams and the extended sequence diagram all TAO entities, their properties, their relationships, their interactions and the internal execution of them can be modeled. Several case studies were modeled using MAS-ML. In particular, three benchmark MAS applications were developed: virtual marketplace, web-based paper submission and reviewing system and a supply chain management system.

In order to promote the widespread use of MAS-ML, a MAS-ML modeling tool needs to be created. The tool should assist the developer of multi-agent systems when modeling and implementing their applications. The goals of the modeling tool are to simplify and accelerate the designs of MAS-ML diagrams. The MAS-ML tool should be composed of an environment for modeling MAS-ML diagrams and a generator machine from the MAS-ML diagrams to code. The viability of such machine was demonstrated by a transformer created to generate code from the structural diagrams.

While creating the MAS-ML language only the UML class and sequence diagrams were analyzed and extended to incorporate MAS characteristics. In order to model different views of a MAS application and to better represent some MAS characteristics other UML diagrams should be analyzed and probably extended. For instance, the UML activity diagram could be extended to also document the logic of a plan and the logic of an action.

## References


[1] Caire, G; Chainho, F.; Evans, R. Agent-oriented analysis using Message/UML. In: Agent-Oriented Software Engineering, Wooldridge, M.; Weiss, G.; Ciancarini, P.(Eds). Second International Workshop, Stringer, 2002.

[2] Carley, K. Computational organizational theory. In: Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence. MIT Press. 1999.

[3] d'Inverno, M.; Luck, M. Understanding Agent Systems. Springer. 2001.

[4] Ferber, J.; Gutknecht, O.; Michael F. From Agents to Organizations: an Organizational View of Multi-Agent Systems. In: Proceeding of the Fourth International Workshop on Agent-Oriented Software Engineering (AOSE), Australia, 2003.

[5] Huget, M. Agent UML Class Diagrams Revisited. In: Bauer, B.; Fischer, K.; Muller, J.; Rumpe, B. (Eds.) Proceedings of Agent Technology and Software Engineering, 2002.

[6] Huns, M.; Singh, M. Agents and Multi-agent Systems: Themes, Approaches and Challenges. In: Readings in Agents, Huns, M.; Singh, M. (Eds.), Morgan Kaufmann, 1998.

[7] Jennings, N. On Agent-based Software Engineering. In: Artificial Intelligence, 117/2, 2000.

[8] Kinny, D. The # Calculus: An Algebraic Agent Language. In: Intelligent Agents VIII, Springer, v.2333, p.32-50. 2002.

[9] Kristensen, B. Subject Composition by Roles. In: Proceedings Object-Oriented Information Systems, Australia, 1997.

[10] Kuncak, V.; Lam, P.; Rinard, M. Role analysis. In: Proceedings 29th Annual ACM Symposium on the Principles of Programming Languages, Portland, 2002.

[11] Letier, E.; Lamsweerde, A. Agent-based Tactics for Goal-Oriented Requirements Elaboration. In: Proceedings of International Conference on Software Engineering, 2002.
[12] Odell, J.; Parunak, H.; Fleisher, M. The Role of Roles in Designing Effective Agent Organizations. In: Garcia, A.; Lucena, C.; Zamboneli, F.; Omicini, A; Castro, J., (Eds.) Software Engineering for Large-Scale Multi-Agent Systems. Springer, 2003.
[13] OMG: Object Management Group, http://www.omg.org, 2004.
[14] Shoham, Y. Agent0: A Simple Agent Language and its Interpreter. In: Proceedings of the Ninth National Conference on Artificial Intelligence, p.704–709, 1991.
[15] Silva, V.; Garcia, A.; Brandao, A.; Chavez, C.; Lucena, C.; Alencar, P. Taming Agents and Objects in Software Engineering. In: Garcia, A.; Lucena, C.; Zamboneli, F.; Omicini, A; Castro, J. (Eds.) Software Engineering for Large-Scale Multi-Agent Systems. Springer 2003
[16] Tursli, W; Maibaum, T. Specification of computer programs, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1987.
[17] Unified Modeling Language Specification, version 2.0 http://www.omg.org/uml, 2004.
[18] Wagner, G. The Agent-Object-Relationship Metamodel: Towards a Unified View of State and Behavior. In: Information Systems, v.28, n.5, 2003.
[19] Wooldridge, M.; Ciancarini, P. Agent-Oriented Software Engineering: the State of the Art. In: Ciancarini,P.;Wooldridge,M.(Eds.) Agent-Oriented Software Engineering, Springer 2001
[20] Yu, L.; Schmid, B. A Conceptual Framework for Agent-Oriented and Role-Based Work on Modeling. In: Wagner, G.; Yu, E. (Eds.). Proceedings of the 1st International Workshop on Agent-Oriented Information Systems, 1999.
[21] Zambonelli, F.; Jennings, N.; Wooldridge, M. Organizational abstractions for the analysis and design of multi-agent systems. In: Ciancarini, P.; Wooldridge, M. (Eds.) Agent-Oriented Software Engineering, Springer 2001.