Using the MAS-ML to model a Multi-Agent System

Viviane Torres da Silva Ricardo Choren Noya

PUC-Rio, Computer Science Department, SoC+Agent Group,
Rua Marques de São Vicente, 225 - 22453-900, Rio de Janeiro, RJ, Brazil
{viviane, choren}@inf.puc-rio.br

PUC-RioInf.MCC 24/03 July, 2003

Abstract. The current object-oriented development practice system analysis is documented through UML artifacts such as Class and Sequence diagrams. Since UML is a widely accepted modeling language, it also would be desirable to offer a UML support for the representation of agent-based system analysis artifacts. Although some central UML constructs are suitably matched for agent-based modeling, several improvements must be made to the UML metamodel to achieve this new goal. This paper presents MAS-ML, a UML extension for agent-based system modeling. The use of MAS-ML for modeling agent-based systems is presented with simple illustrating application.

Keywords. Multi-agent system, modeling language, UML, conservative extension

Resumo. O desenvolvimento de sistemas orientados a objetos é atualmente documentado utilizando artefatos de UML como os diagramas de Classe e Seqüência. Devido UML ser uma linguagem de modelagem amplamente aceita na comunidade científica, é desejado utilizar UML para dar suporte a representação dos artefatos de análise de sistemas multi-agentes. Embora alguns conceitos centrais de UML também sejam utilizados na modelagem de sistemas multi-agentes, é necessário estender o meta-modelo de UML para incluir conceitos específicos de sistemas multi-agentes. Neste sentido, o artigo apresenta MAS-ML, uma extensão de UML para modelagem de sistemas multi-agentes. O uso de MAS-ML para modelagem de sistemas baseados em agentes é apresentado utilizando-se uma aplicação ilustrativa.

Palavras-chave. Sistema multi-agente, linguagem de modelagem, UML, extensão conservativa.

1 Introduction

Agent-based systems require adequate techniques to explore their benefits and unique characteristics. In particular, modeling languages for multi-agent systems should explore the use of agents and other abstractions as first order modeling elements. These languages should represent the static and dynamic aspects of such systems by expressing the characteristics of all essential elements of MASs.

Some modeling languages proposed in the literature, such as [8, 12], are based on the UML to reduce the risk of adopting a new technology. The UML is used as a basis for extension because it is widely accepted as a *de facto* standard for object-oriented modeling. We also believe as in [8] that a new modeling language should preferably be an incremental extension of a known and trusted modeling language. Nevertheless, in its original form UML provides insufficient support for modeling MASs. Among other things, the UML metamodel does not provide support for modeling agents, organizations and agent roles. Current approaches [8,12] propose to extend UML by expressing agents as a stereotype of objects/classes. This is not satisfactory because stereotypes can only be used to indicate a difference in meaning or usage between two model elements with identical structures. Based on the definition presented in the UML specification [7], stereotypes may not be used to represent two completely different abstractions. As a consequence, the current UML extensions to deal with the fundamental characteristics of MAS neither define nor clearly represent the elements they identify nor what are their relationships. Therefore, we felt the need for a new approach to extend the UML metamodel so that it could incorporate all features of multi-agent systems.

The MAS-ML modeling language (Multi-Agent System Modeling Language) [10] was developed by extending the UML based on the TAO (Taming Agents and Objects) conceptual framework (metamodel) [9]. TAO provides an ontology that makes it possible to understand the abstractions, and their relationships, used to support the development of large-scale MASs. The ontology associates well-accepted abstractions, such as objects and classes, with other abstractions, such as agents, roles and organizations.

The UML metamodel was extended by preserving all object-related concepts while including agent-related ones. To extend the UML metamodel according to the TAO metamodel concepts, we needed more than the three extension mechanisms provided by the UML, namely: tag definitions, constraints and stereotypes. Often it was necessary to add new metaclasses to the UML metamodel to represent some new MAS concepts.

MAS-ML uses UML as a general modeling platform. MAS-ML's purpose is to offer extensions to the UML metamodel in order to incorporate new agent modeling capabilities, thus being a concrete proposal for improving current versions of the UML. This paper provides an example of MAS-ML to illustrate the impacts of the proposed extensions on an agent-based system modeling application. The difference between our approach and others described in the literature is the clear definition and representation of the elements that compose MASs and their behavior.

The paper is structured as follows. Section 2 briefly presents MAS-ML and the TAO conceptual framework. Section 3 describes a modeling approach used to perform the system design using the MAS-ML. Section 4 defines the virtual marketplace example and presents the models generated using the MAS-ML. Section 5 provides some comparisons between our approach and related work. Finally, Section 6 offers some conclusions and highlights some ongoing work.

2 MAS-ML

MAS-ML is a modeling language that makes additive extensions to UML to provide support for multi-agent systems modeling. By augmenting the UML metamodel, new modeling capabilities were incorporated into UML, contributing to its evolution. The UML metamodel extensions proposed by the MAS-ML are based on the concepts defined in the TAO.

2.1 The TAO Metamodel

The TAO metamodel provides an ontology to capture the foundations for agent and object-based software engineering. Hereafter, we introduce the static and dynamic aspects of the metamodel, which are related to MAS-ML features in this paper; see [10,11] for more details.

2.1.1 TAO Static Aspects

The static aspect of the TAO metamodel captures a rich variety of concepts involved in the structure of a multi-agent system, namely, object, agent, organization, object role, agent role, environment and event.

- *Object*: an object is a passive element in the domain whose instances have state and behavior. An object may evolve from state to state. However, it has no control over its behavior, meaning that it does whatever any other element asks it to do and only when it is asked.
- *Agent*: an agent is an autonomous, adaptive and interactive element in the domain whose instances are expressed through mental components such as beliefs, goals, plans and actions. An agent acts as a processor for plans of actions that are executed, using its beliefs, to accomplish a goal. A goal is an objective the system should meet, and a belief is some knowledge about the system.
- *Organization*: an organization is an element in the domain whose instances group agents, objects and other organizations (sub-organizations). An organization has goals, beliefs (as agents) and axioms. An axiom characterizes global constraints that agents must obey. An organization also is responsible for defining the roles that will be played by agents, sub-organizations and objects. At least one organization must inhabit the environment. We call this organization main-organization.
- *Object Role*: an object role guides and restricts the behavior of an object through the description of a set of features that are viewed by other elements. An object role may restrict access to the state and behavior of an object instance, but may also add information, behavior and relationships to the object instance that plays the role.
- *Agent Role*: an agent role guides and restricts the behavior of an agent through the description of a set of goals, beliefs and actions. An agent role defines duties, rights and protocols that restrict an agent instance. A duty defines an action that must be executed by an agent instance; a right defines an action that may be executed by an agent instance; and a protocol defines an interaction between an agent role and other elements.
- *Environment*: an environment is an element in the domain whose instances are the habitat for agents, objects and organizations. An environment can be passive as an object, or active as an agent.
- *Event*: an event is an element whose instances are generated by other elements in the domain. An event can be generated by objects through the execution of their operations, by agents through the execution of their actions and by the environment when the environment is an active element.

The TAO metamodel defines relationships in which the above concepts may be involved. These are:

- *Inhabit*: specifies that the element that inhabits is created and destroyed in the habitat and may leave and enter habitats, respecting the habitat permissions. Inhabit is applied to environments and agents, environments and objects and environments and organizations.
- *Ownership*: specifies that an element is defined in and must obey a set of constraints defined by another element. The member element does not exist outside of the scope of its owner. Ownership is applied to roles (members) and organizations (owners).
- *Play*: specifies that an element that plays a role assumes its properties and relationships. The behavior of the element is guided by and restricted to the scope of the role. Every agent or sub-organization plays at least one role in an organization. Objects also can play roles.
- *Specialization (Inheritance)*: defines that a sub-element that specializes a super-element inherits all the state and behavior associated with the super-element. A sub-element also may add and redefine the properties and behavior associated with the super-element. Specialization may be used between objects, agents, organizations, object roles and agent roles.
- *Control*: defines that a controlled element must do anything that a controller element requests. An agent role can control another agent role or an object role. Object roles only can control another object role.
- *Dependency*: defines that an element (client) may be defined to be dependent upon another one (supplier) to do its job. In other words, that the client cannot completely do its job unless it asks the

supplier. An agent role can depend on another agent role and an object role can depend on another object role.

- *Association*: defines how one element interacts with another, indicating that these elements know each other. Associations may be used between (i) roles (object or agent roles), (ii) environments, (iii) objects, (iv) agents and objects and (iv) organization and objects.
- *Aggregation*: defines that an element is part of an aggregator. The aggregator may use the functionality available in its parts. This relationship may be applied between object roles, agent roles, objects, agents and organizations.

2.1.2 TAO Dynamic Aspects

The dynamic aspects of the TAO metamodel describe the relationships between its static elements. They can be classified as primitive (elementary) dynamic processes and high-level dynamic processes.

Primitive processes describe the most basic domain-independent interactions that exist between elements. The processes of creating and destroying the elements of an MAS are characterized as primitive processes. These processes define the actors, preconditions, execution steps and post conditions involved. They encompass processes for object, object role, agent, agent role, organization and environment creation and destruction [11].

High-level dynamic processes are more complex domain-independent behavior that are described based on primitive and other high-level dynamic processes. They derive from the characteristics of the relationships between entities that are associated with domain-independent behavior: ownership, play and inhabit relationships. They encompass processes for an agent entering or leaving an organization, an organization entering or leaving another organization, and an agent or an organization entering or leaving an environment.

2.2 MAS-ML Static Aspects

The definitions of the object and event concepts of the TAO metamodel are similar to those of the Class and Event UML meta-classes; thus their notation is preserved in MAS-ML. However, it is necessary to create new meta-class definitions for the other TAO concepts. MAS-ML has included the AgentClass, OrganizationClass, EnvironmentClass, ObjectRoleClass and AgentRoleClass meta-classes to the UML metamodel [10]. Each meta-class has a corresponding notation in MAS-ML. Similar to the diagram element that defines a class in UML, the new diagram elements are composed of three compartments (UML metamodel terminology) separated by horizontal lines. The top compartment holds the class name that must be unique in its enclosing namespace. The middle list compartment holds the list of structural features and the bottom list compartment holds a list of behavioral features. Either or both of the middle and bottom compartments may be suppressed when necessary. For example, the diagram element that models an AgentClass defines the name of the agent in the first compartment, the goals and beliefs of the agent in the middle compartment and its plans and actions in the bottom compartment. Furthermore, new diagram elements have been created and associated with new relationships defined in TAO that do not exist in UML. These relationships are Inhabit, Ownership, Play and Control.

MAS-ML extends the Class diagram to include modeling information about agents. This diagram shows the association, aggregation and the specialization relationships of TAO. Moreover, MAS-ML defines two other diagrams: Organization and Role diagrams.

The Organization diagram models the system organizations identifying their habitats, the roles that they define and the elements – objects, agents and sub-organizations – that play those roles. This diagram shows the ownership, play and inhabits relationships of TAO. The Role diagram is responsible for clarifying the relationships between the agent roles and object roles. This diagram shows the control, dependency, association, aggregation and specialization relationships of TAO.

2.3 MAS-ML Dynamic Aspects

MAS-ML proposes an extension to the UML Sequence diagram to model the dynamic aspects based on the TAO metamodel. MAS-ML extended Sequence diagram has three new elements to represent the agent, the organization and the environment concepts. Furthermore, it proposes new ways to define pathnames that identify element instances. Sequence diagrams illustrate (i) agents and organizations committing to roles and changing their roles, (ii) agents and organizations sending and receiving messages, (iii) agents and organizations executing actions, (iv) elements calling methods of an object, (v) objects executing methods and (vi) the creation and destruction of elements.

To model the dynamic processes presented above, MAS-ML extended Sequence diagram defines new stereotypes and extends the definition of the existing UML <<create>> and <<destroy>> stereotypes. The new stereotypes are: <<role_commitment>>, <<role_cancel>>, <<role_change>>, <<role_activate>> and <<role_deactivate>> [11].

- *create*: this stereotype was specialized to represent the creation of agents, organizations and environments; the association of a role instance to agent and organization instances; and the creation of an object and the association of a role to the object.
- *destroy*: this stereotype was specialized to represent the destruction of agents, organizations and environments, and the destruction of all role instances associated with agents, organizations and objects.
- *role_commitment*: this stereotype was created to represent an agent, organization or object committing to a role. It can represent an agent or an organization entering an organization to play a role but cannot represent an agent or an organization entering a new environment.
- *role_cancel*: this stereotype represents that a role is being canceled, i.e. an element stops playing the role. It also is used to illustrate when an agent or an organization leaves another organization.
- *role_deactivate*: this stereotype changes the state of a role that an agent or organization is playing from active to inactive.
- *role_activate*: this stereotype changes the state of a role from inactive to active.
- *role_change*: this stereotype represents an agent or an organization changing its role. An object does not change from one role to another because it does not have the autonomy to choose its roles.

3 The Modeling Approach

This section presents a modeling approach to guide the understanding of the use of MAS-ML for modeling a realistic example presented in Section 4. The modeling approach describes the elements that must be defined in order to create each MAS-ML diagram. The static aspects of an application are modeled using the three static diagrams proposed by MAS-ML – Organization, Role and Class diagram. The dynamic aspects of an application are modeled using the dynamic Sequence diagram.

Organization and Role diagrams

An Organization diagram models an environment, an organization, its sub-organizations, the roles defined in the organization and the elements that play those roles. Therefore, an Organization diagram is completely modeled when (i) the environment and the main-organization have been defined, (ii) the object roles and the agent roles defined in the main-organization have been identified, (iii) its sub-organizations have been defined and (iv) the agents and objects have been identified.

However, the Organization diagram does not describe the relationships between the roles. The Role diagram complements the Organization diagram by modeling the relationships between object roles and agent roles. A Role diagram is completely modeled while all the object roles and agent roles played by objects, agents and sub-organizations have been defined.

Environment and main-organization identification: It is important to analyze the environment characteristics while defining it. If the environment is a passive element it should be modeled as an object and its attributes and methods should be defined. However, if the environment is an active element it should be modeled as an agent and its goals, plans, actions and beliefs should be specified.

The definition of a main-organization comprises the specification of its goals, plans, actions, beliefs and axioms. It also is necessary to define the roles defined in the organization according to the definition of the main-organization axioms.

Roles identification: An object role is defined by its attributes and methods. An object role can add attributes to the object that play the role and can also restrict the access other attributes defined by the object. The object role can add some methods and can also restrict the access to other methods defined by the object. An agent role is defined by its goals, duties, rights and protocols. The goals should be analyzed before other properties since they influence these properties. Protocols specify messages sent and received when two associated roles are interacting. During the protocols analysis, the duties and rights that agents and sub-organizations must obey while playing the role may also be detailed. Protocols, duties and rights must obey the axioms specified in the organization.

Sub-organizations identification: The agent roles previously defined can be so complex that they should be played by sub-organizations. If there are sub-organizations, they must be defined; i.e., the goals, plans, actions, beliefs and axioms must be specified. Its roles must also be identified for each sub-organization. Each sub-organization will have its own Organization diagram.

Class diagram
A Class diagram models the relationships between objects, agents, organizations and environments. A Class diagram is completely modeled when (i) the environment has been identified, (ii) the main-organization and all its sub-organization have been defined and (iii) the agents and objects have been identified.

Agents and objects identification: An object is defined through the specification of its attributes and methods. The definition of an agent involves the specification of its goals, plans, actions and beliefs. The roles associated with an agent influence its definition. The goals of the roles that an agent plays are related to the goals of the agent. The actions and plans of the agent are directly associated with the duties, rights and protocols defined in the roles. For instance, some actions must be associated with the ability of sending and receiving the messages described in protocols.

Sequence diagram
A Sequence diagram models the interactions between (i) agents playing roles, (ii) organizations playing roles, (iii) environments and (iv) objects while either playing roles or not. Therefore, the Sequence diagram depends on the identification of all elements that are defined in the MAS application.

4 Modeling Example

A virtual marketplace was modeled to illustrate the use of the MAS-ML in a practical application. An electronic commerce example was chosen since it is referred to in the literature [4, 6] as an appropriate example of a multi-agent system.

The *virtual marketplace* is composed of a *main-market* where users are able to negotiate any type of item. In addition, the *main-market* defines two market types that negotiate items with particular characteristics. The *markets of special goods* negotiate expensive high-quality items and the *markets of used goods* negotiate low-priced low-quality items. Users can buy items in the *main-market*, in *markets of special goods* or in *markets of used goods*. The users can also sell their items in the *used goods*. In the *main-market* and in *markets of special goods* users buy the items available in the market.

In the *main-market* and in *markets of special goods*, the user looks for a seller and sends it a description of the desired item. The seller, created by the market to negotiate with the buyer, is responsible for verifying if there is an item with the same characteristics in the *environment* (the virtual marketplace). The *environment* stores all the items to be sold in these markets. If the item is found, the seller negotiates with the buyer.

In *markets of used goods*, the sellers and the buyers are users. The users that want to sell items announce them. And the users that want to buy items look for announcements in the market. If the buyer finds the desired item, it starts a negotiation with the seller.

The marketplace example will be expressed following the modeling approach defined in Section 3. The identification of the MAS elements presented in the example will be defined in order to create static and dynamic diagrams.

4.1 Static diagrams

The organization modeled presented in

Figure 1 illustrates the main-organization. In order to create this diagram the environment and the main-organization were defined together with the roles, the sub-organizations and the elements that play the roles.

Organizations are shown as a lozenge shape (a symbol with horizontal sizes and convex top and bottom). The diagram presents the main-organization *General Store* and two sub-organizations, *Store of Imported Books* and *Second-hand Bookstore*. Agents are represented as a rounded rectangle and the agent roles as a solid rectangle with a curve on its bottom. Two agent types were modeled in this system, *user agent* and *store agent*. The diagram also illustrates the *seller* and *buyer* roles defined by the main-organization and played by *store agents* and *user agents*, respectively.

The object roles are represented as a solid rectangle with an angle in its left corner. The diagram shows the object *item* and the *desire* and *offer* roles that it can play. The *ownership* relationship between the organization and its roles is shown as a double line linking the owner (main-organization) to the member (each role). The *play* relationship between an element and a role is shown as a simple line linking the element that plays the role to the double line that describes the ownership relationship.

The environment *Virtual Marketplace*, modeled as a passive element, is shown as a package that brings together all the entities that inhabit it. The inhabit relationship is shown by inserting the citizen in the lower compartment defined by the diagram element associated with the environment. All diagram elements were illustrated using the simplified representation that suppresses the middle and the bottom compartments.

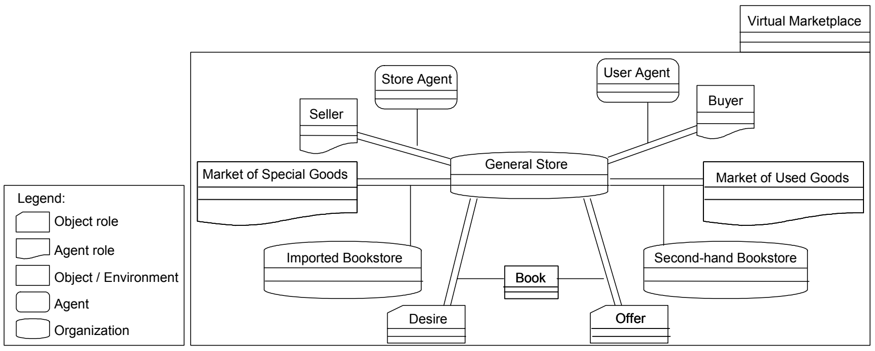


Figure 1 – Organization diagram modeling the main-organization

Environment and main-organization identification:
From the problem description, it is possible to identify a main-organization inhabiting the *Virtual Marketplace* environment. The environment is a passive element modeled as an object that stores items to be negotiated as one of its *attributes*. It implements the get and set *methods* to access these items. Since the user can move from one marketplace to another, the environment may also store information about other environments.

The main-organization *General Store* will represent the system main-market. As users can buy items in the main-market, the main-organization defines the *buyer* and *seller agent roles.* These roles will be detailed below. The main-organization *goal* is the management of sellers and orders. To achieve its goals, the main-organization defines *plans* to (i) create sellers to negotiate with buyers, (ii) to update the environment to inform that an item is no longer available and (iii) to evaluate the profit that results from sales. To guarantee that the main-organization will receive the information about all sales, an *axiom* is defined. The main-organization *beliefs* are related to the information about buyers, sellers and sales.

Roles identification:
The main-organization defines *buyers* and *sellers* whose *goals* are to buy and sell items, respectively. To achieve these goals, they negotiate items stored in the environment. The buyer and seller roles define a "simple negotiation" *protocol* that describes how the elements playing these roles should interact. This protocol defines that a buyer asks a seller the price of an item. After consulting the environment, the seller sends the price to the buyer and the buyer can accept or reject the seller proposal. The choice of accepting or rejecting a given seller proposal is one of the buyer role *rights*. If the buyer accepts the seller proposal, the seller sends the bill to the buyer. Then, the seller sends the information about the sale to the main-organization, as specified in its axioms. This characterizes a *duty* of the seller.

Buyers and sellers have different views of the items they negotiate. An *item* is a *desire* to buyers and an *offer* to sellers. The *desire* and *offer* object roles have different characteristics. Suppose the item negotiated in the market place is a book. A book has a set of attributes (title, author, ISBN, price) and a set of methods (getters and setters to each attribute). The desire role allows the buyer to set the title, author and ISBN of a book, but it only allows the buyer to get the price. Since the environment stores items to sell and informs the seller about these items, the seller only needs to manipulate the price attribute. The offer role allows the seller to get the price of an item since the seller must generate a bill based on it.

The main-organization defines the *market of special goods* and *market of used goods* roles played by sub-organizations. Since the markets of special goods sell expensive goods, these markets check if the users that want to enter the market can afford the items they want to by. The management of new buyers is one of the market's *goals*. Another goal is to manage the creation of sellers when buyers enter the market.

Since the role of the market of special goods is defined in the scope of the main-organization, it must obey its *axioms*. Moreover, to guarantee that its sellers will send it the information about the sales, the market of special goods also defines an *axiom*. In addition, the market of special goods defines *protocols* to guide the interactions (i) between itself and new buyers, (ii) between itself and its own buyers and (iii) between itself and its own sellers.

In the market of used goods, users can buy and sell items. The market does not control sellers since they represent users selling items that are not stored in the environment. A seller announces items in the market and buyers search for these announcements. The market of used goods does not define any restriction to the entrance of new buyers or new sellers.

The market of used goods also has to follow the axioms defined in the main-organization (to send information about the sales). The market of used goods' *goals* are to manage announcements and to evaluate profits. The market of used goods defines *axioms* to guarantee that its sellers will send an amount of money related to sales. The *protocols* that the market of used goods defines are related to the interactions (i) between itself and new buyers, (ii) between itself and its sellers and (iii) between itself and its buyers.

Sub-organizations identification:
Examples of organizations that may play the market of special goods role are imported bookstores and exquisite goods stores. Examples of organizations that may play a role as markets of used goods are second-hand clothing stores and second-hand bookstores. In this paper we will model the imported bookstore and second-hand bookstore. These organizations define *buyers* and *sellers* that are different from those previously defined in the main-organization, namely, *buyers* and *sellers of imported books* and *buyers* and *sellers of second-hand books*, respectively.

Since the imported bookstore plays the role of a market of special goods, its goals must be compatible. The *goals* of the imported bookstore are to manage the inclusion of new buyers in the market and to manage the creation of sellers. To achieve its goals it defines *plans* (i) to negotiate the entrance of a new buyer, checking if the buyer has the needed characteristics (according to the *protocol* that defines the interaction between itself and a buyer), (ii) to register the new buyer (according to the *protocol* that defines the interaction between itself and a buyer of imported books) and (iii) to create the seller of imported books (according to a *protocol* that defines the interaction between itself and a seller of imported books). It also defines a *plan* to receive sales information (according to the *protocols* defined between itself and its sellers and according to the *axiom* defined in the market of special goods). Then, it sends information to the main-organization by following the *duties* specified in the role. The *beliefs* of the imported goods store are its buyers, sellers, sales and the main-organization.

The *goods* of the second-hand bookstore (which plays the market of used goods role) manage the announcement of the items to be sold and evaluate profits. The second-hand bookstore defines *plans* (i) to store the announcements, (ii) to send them to buyers and (iii) to receive the monies generated by the sales. Its *beliefs* are related to the sales and to the announcements.

Figure 2 presents an Organization diagram modeling the *second-hand bookstore*. For space reasons, this article will not present the Organization diagram of the sub-organization of the imported goods store.

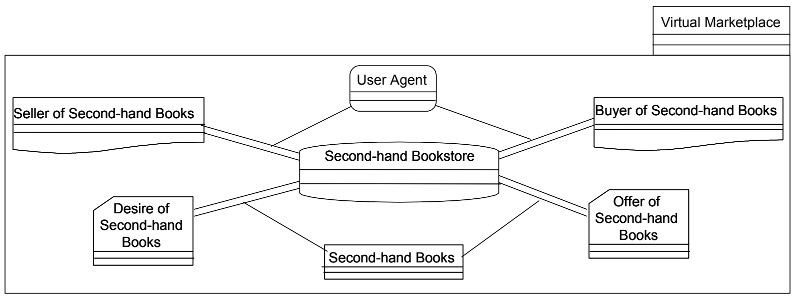

Virtual Marketplace
User Agent
Seller of Second-hand Books
Buyer of Second-hand Books
Second-hand Bookstore
Desire of Second-hand Books
Offer of Second-hand Books
Second-hand Books

Figure 2 – Organization diagram modeling the second-hand bookstore

Roles identification:

(i) Roles played by agents and objects in the main-organization:

As we have already seen, the main-organization defines the *buyer* and *seller* roles whose *goals* are to buy and sell items, respectively. To achieve these goals, they negotiate items stored in the environment. The buyer and seller roles define a "simple negotiation" *protocol* that describes how the elements playing these roles should interact. This protocol defines that a buyer asks a seller the price of an item. After consulting the environment, the seller sends the price to the buyer and the buyer can accept or reject the seller proposal. The choice of accepting or rejecting a given seller proposal is one of the buyer role *rights*. If the buyer accepts the seller proposal, the seller sends the bill to the buyer. Then, the seller sends the information about the sale to the main-organization, as specified in its axioms.

Buyers and sellers have different views of the items they negotiate. An *item* is a *desire* to buyers and an *offer* to sellers. The *desire* and *offer object roles* have different characteristics. Suppose the item negotiated in the market place is a book. A book has a set of attributes (title, author, ISBN, price) and a set of methods (getters and setters to each attribute). The desire role allows the buyer to set the title, author, and ISBN of a book, but it allows the buyer to get only the price. Since the environment stores items to sell and informs the seller about these items, the seller only needs to manipulate the price attribute. The offer role allows the seller to get the price of an item since the seller must generate a bill based on it.

(ii) Roles played by agent and objects in stores of imported books and in second-hand bookstores

These organizations also define *buyers* and *sellers*. However, they are different from those previously defined in the main-organization, namely, *buyers* and *sellers of imported books* and *buyers* and *sellers of second-hand books*, respectively. All the buyers and the sellers in the system have the same *goals*: to buy and to sell an item. However, they define different *duties*, *rights* and *protocols*. In order to generate the Role diagram illustrating the relationships between the roles, the protocols that they defined are described.

The *buyer* defined in the main-organization defines the "simple negotiation" protocol with the seller and the "entering organization" protocol with both the market of special goods and market of used goods. The *buyer of imported books* specifies (i) the "registration" protocol with the markets of special goods, (ii) the "searching for seller" protocol with the market of special goods and (iii) the "simple negotiation" protocol with the seller of imported books. Finally the *buyer of used books* defines (i) the "registration" protocol with the market of used goods, (ii) the "simple negotiation" protocol with the seller of used books, (iii) the "complex negotiation" protocol with and the seller of used books and (iv) the "searching for announcement" with the market of used goods.

The *seller* defined in the main-organization only specifies the "simple negotiation" protocol with the buyer. The seller role is extended by the *seller of imported books* because the seller of imported books also defines the "simple negotiation" protocol. The seller of imported books defines the "simple negotiation" protocol with the buyer of imported books and the "register sale" protocol with the market of special goods. Moreover, the *seller of second-hand books* extends the seller of imported books role additionally defining the "complex negotiation" protocols with the buyer of used books and the "announcing" protocol with the market of used goods.

Since the items sold in imported bookstores and in second-hand bookstores are different it is necessary to define new objects to represent them. The *imported book* and the *second-hand book* have the same properties associated with an item and additional attributes to indicate the item's country of origin and an attribute to indicate the item's appearance. New desire and offer object roles should be defined due to the creation of these new objects. The *desire of second-hand books* and *offer of second-hand books object roles* extend the *desire* and *offer* roles, respectively, including methods to access the origin country attribute. The *desire of imported books* and *offer of imported books* roles extend the *desire* and *offer* roles, respectively, including methods to access the appearance attribute. Figure 3 illustrates a Role diagram that emphasizes the relationship between agent roles and object roles (Part I) and also the specialization relationship between object roles and agent roles (Part II).

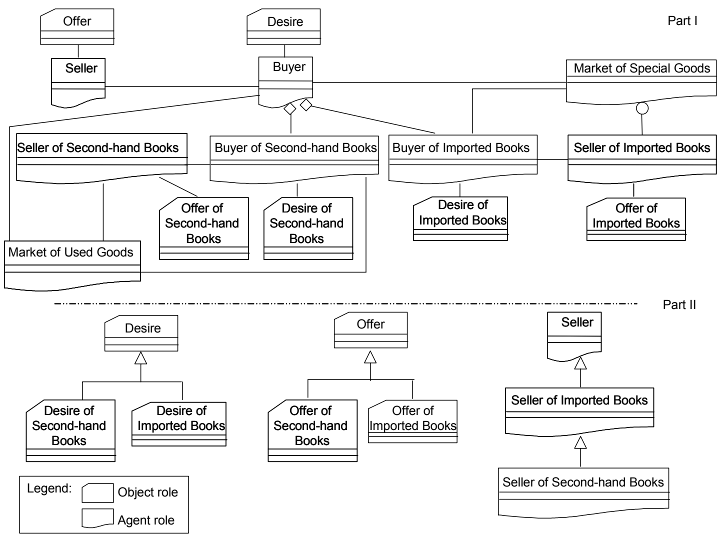


Figure 3 – Role diagram

Agents and classes identification:
As described before, the items being sold in the organizations are books. There are three kinds of books: simple book, imported book and second-hand book. Imported books and second-hand books have similar properties described in the simple book class. Additionally, the imported book class defines an attribute to describe the item origin country and the second-hand book class defines an attribute to indicate the item appearance.

At this point, it is also necessary to describe the agents that will play the agent roles. As already mentioned, there are two different kinds of agents. The *user agent* represents the users in the system. A user agent is created when a new user wants to buy or sell an item. The user agent *goals* depend on the user goals. To achieve its *goals*, the user agent may play the *roles* of buyer, buyer of imported books, buyer of second-hand books and seller of second-hand books. The user agent may have *plans* associated with its roles and goals. The user agent may, for instance, define *plans* to buy an item in the main-market and to sell an item in the second-hand bookstores. Plans are related to protocols defined in the role the agent is playing and they respect the duties and rights defined in this role. The user agent *beliefs* store the user preferences and information related to the agent experience.

The *store agent* represents the system preferences and it is created whenever a user wants to buy an item. The *store agent* has a unique *goal;* that is, to sell an item. The agent can play the seller and seller of imported books *roles*. Independently from the role that the agent is playing, it executes the same *plan* associated with its unique goal.

The Class diagram illustrated in Figure 4 completes the modeling of the system static aspects. The Class diagram specifies the relationships among objects, among organizations and among environments. The environment has a self-reference relationship since agents can move from one environment instance to another. Agents are not modeled in this diagram because they do not directly interact with objects, organizations and other agents. All interaction takes place through their roles. The imported book and second-hand book objects specialize the book object. The store of goods organization groups the store of imported books and the second-hand bookstore organizations.

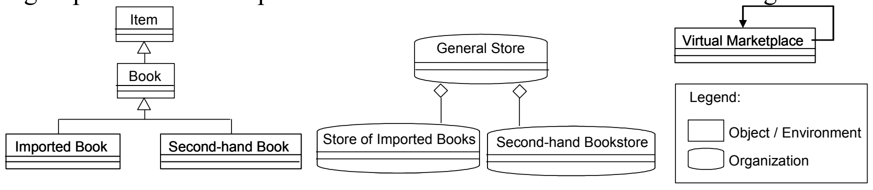


Figure 4 – Class diagram

4.2 Dynamic diagrams

The dynamic aspects of MAS model the interactions between agents, organizations, environments and objects. To illustrate the use of the MAS-ML Sequence diagram, two dynamic features are modeled: (i) a user agent, playing the buyer role, entering a second-hand bookstore to negotiate with a seller of second-hand books (Figure 5); and (ii) a user agent moving from one environment to another to play a buyer of second-books role (Figure 6). The user agent has the ability to move from one organization to another and to move from one environment to another as specified in the problem definition.

In Figure 5 the user agent instance called *Bob* moves from the main-organization *Wall-Market* to a second-hand bookstore called *Siciliano*. These two organizations inhabit the same virtual marketplace called *Place I*. The user agent interacts with the second-hand bookstore *Siciliano* to check if its goals are compatible with the goals of the roles defined in the organization. The agent requests permission to play the buyer of second-hand books role. Second-hand bookstores always allow an agent to enter and play one of its roles. The agent changes its role by canceling the buyer role and creating the buyer of second-hand book role. The act of changing roles is represented by the stereotype `<<role_change>>`. The agent playing the buyer of second-hand books role requests the announcement of an item. The organization sends the announcement to the buyer and the buyer negotiates the item with the seller.

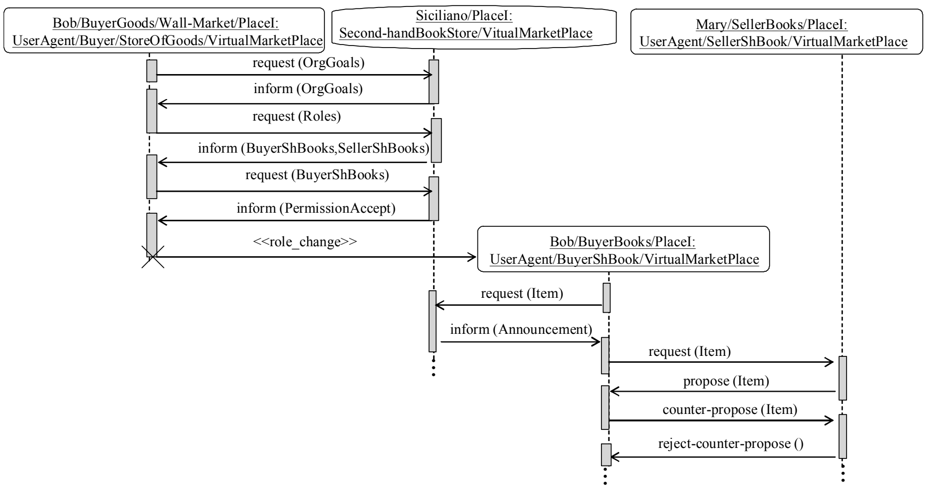
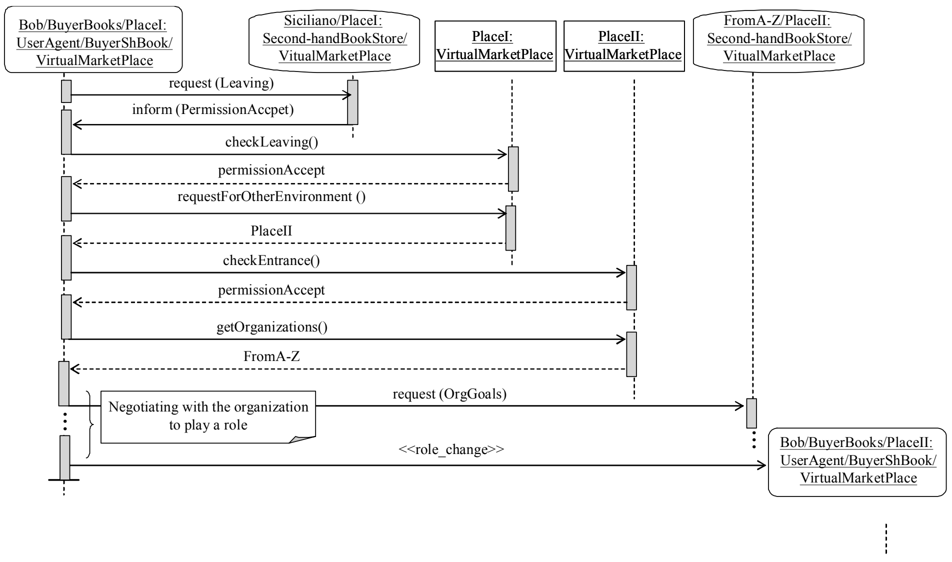

Bob/BuyerGoods/Wall-Market/PlaceI:
UserAgent/Buyer/StoreOfGoods/VirtualMarketPlace
Siciliano/PlaceI:
Second-handBookStore/VitualMarketPlace
Mary/SellerBooks/PlaceI:
UserAgent/SellerShBook/VirtualMarketPlace
request (OrgGoals)
inform (OrgGoals)
request (Roles)
inform (BuyerShBooks,SellerShBooks)
request (BuyerShBooks)
inform (PermissionAccept)
<<role_change>>
Bob/BuyerBooks/PlaceI:
UserAgent/BuyerShBook/VirtualMarketPlace
request (Item)
inform (Announcement)
request (Item)
propose (Item)
counter-propose (Item)
reject-counter-propose ()

Figure 5 – Sequence diagram (Part I)

The shapes that represent the elements in the MAS-ML Sequence diagram are similar to the shapes that represent the elements in the static diagrams. Objects are shown as rectangles, agents are shown as rounded rectangles and organizations are shown as lozenge shapes. The environment is also shown as a rectangle because it is a passive element. Furthermore, to completely specify the elements participating in an interaction we must mention the pathnames associated with them. Different elements have different pathnames [11]. For instance, the pathname that describes an agent specifies (i) the agent instance, (ii) the role instance that the agent is playing, (iii) the organization hierarchy where the agent is playing the role, (iv) the environment that it inhabits and (v) their corresponding class names. The information about the role, organizations and environment can be suppressed and a simple pathname can be used.

Suppose the buyer has tried to negotiate with all sellers and that *Siciliano* is the only second-hand bookstore in *Place I*. The buyer alternative is to move to another environment to try to buy the item. Figure 6 demonstrates the user agent *Bob* moving from the environment *Place I* to *Place II*.

Before moving from one environment to another, the agent must check whether it can leave the organization where it is playing roles and can leave its current environment. Then, it requests to its current environment the address of another environment. The agent interacts with the other environment to check if it can enter it. After the user agent is accepted by the environment, it requests an organization to play a role. The process of choosing a role to play in an organization already has been explained in the description of Figure 5. After choosing the role, the agent moves from the organization *Siciliano* in Place I to the organization *From A-Z* in *Place II*. The agent deactivates the buyer role and creates the buyer of second-hand book role.



Bob/BuyerBooks/PlaceI: UserAgent/BuyerShBook/ VirtualMarketPlace
Siciliano/PlaceI: Second-handBookStore/ VitualMarketPlace
PlaceI: VirtualMarketPlace
PlaceII: VirtualMarketPlace
FromA-Z/PlaceII: Second-handBookStore/ VitualMarketPlace
request (Leaving)
inform (PermissionAccpet)
checkLeaving()
permissionAccept
requestForOtherEnvironment ()
PlaceII
checkEntrance()
permissionAccept
getOrganizations()
FromA-Z
request (OrgGoals)
Negotiating with the organization to play a role
<<role_change>>
Bob/BuyerBooks/PlaceII: UserAgent/BuyerShBook/ VirtualMarketPlace

Figure 6 – Sequence diagram (Part II)

In order to simplify the modeling example, we chose not to represent the actions or the internal methods executed by the elements in the Sequence diagram. However, the Sequence diagrams represent the messages being sent by elements and the method being called by elements.

5 Discussion and Related Work

The virtual market example presented in Section 4 illustrated the use of the MAS-ML modeling language. The Organization diagram was used (i) to present the organization and its roles, (ii) to describe the agents and objects that play their roles and (iii) to define the environment that the elements inhabit. The Role diagram illustrates all the agent and object roles defined in the example and their relationships. The Class diagram presents the system's objects and the relationships of the organizations. The Sequence diagram was used to illustrate two dynamic processes, one representing an agent entering an organization to negotiate and another representing an agent changing its environment. The diagrams show the interaction between agents, organizations, environments and objects. They illustrate (i) the creation of a role instance, (ii) agent commitment to this role, (iii) the cancellation of the previous role of the agent, (iv) agents and organizations sending and receiving messages and (v) agents calling environment methods.

For obvious space reasons the example could not explore all the potential of the proposed modeling language, it presents the essential features of the structural and dynamic aspects. Using the Organization, Role and Class diagrams proposed in the MAS-ML it is possible to model the essential static aspects. The MAS-ML Sequence diagrams allow for the expression of the complex dynamic characteristics of MASs as seen in the example. This is so because agents, organizations and roles are used as first order elements in the diagrams.

MAS-ML has been used by the authors to express more complex versions of the present example in our laboratory. We are in the process of supervising the use of MAS-ML in number of different MAS applications. Reports of these experiments will be available by the time this book is published. The first practical conclusions point to the fact that a design requires considerable MAS expertise to be able to take advantage of all the language features. Tool support will be a key resource to make MAS-ML dissemination viable.

The main difference between the approach presented in this work and other approaches is the clear definition and representation of the elements that compose MASs. AUML [1, 5, 8] is a UML extension for MAS proposed by FIPA [3]. AUML does not make a clear distinction between the agent and the object concepts and their respective representations. In addition, AUML does not describe, in its models, inherent MAS entities such as environment and organization. As a consequence, AUML does not allow the specification of the static relationships between organizations, roles and agents (play) or the static relationships between agents and environments (inhabit). Therefore, the dynamic behavior of an agent playing more than one role in different organizations and moving from one environment to another cannot be modeled.

Other initiatives such as [2, 12] also propose UML extensions for MAS. AORUML [12, 13] does not describe the relationships among agents, objects and other MAS entities. MESSAGE [2] creates specific diagrams to represent agents and other MAS elements, adding these diagrams to the set of UML diagrams. Although these diagrams model elements such as organizations and roles, none of them present the dynamic interactions among MAS elements.

6 Conclusions and Ongoing Work

Rich and precise representation of concrete problem domains, using an agent-oriented modeling language, is an important success factor for agent-based development. This is important to ensure that agent-oriented methodologies will indeed be able to be in the mainstream for the development of information systems.

MAS-ML is a modeling language based on the definition of the structural and dynamic aspects presented in the TAO conceptual framework. It extends UML, introducing meta-classes, stereotypes and diagram elements to represent MASs concepts. MAS-ML extensions adapt two existing UML diagrams and create two other diagrams that display new elements, such as agents, organizations, roles and environments using different icons [10].

This paper presented an example to demonstrate the usefulness of MAS-ML. The example followed a modeling approach to guide the design method using MAS-ML and to illustrate the relevance of the proposed extensions. While further case studies and examples are needed to test and refine the language, the use of MAS-ML has demonstrated that it is a practical modeling language to support agent-oriented software specification.

It is important to mention that there is no widely accepted programming language that considers agents as first order abstractions. Software development based on agent-oriented paradigm depends on programming languages so that it may be possible to evaluate the traceability between the requirement analysis and the implementation code [8]. To fill this gap, our present ongoing work aims at mapping the design produced using the MAS-ML to object-oriented code. This can be used to automatically generate code from an MAS-ML model.

As already mentioned, MAS-ML has a direct impact on two UML diagrams. Another interesting path for further work is to assess how the proposed metamodel extensions will influence other UML diagrams.

Acknowledgement. This work has been partially supported by CNPq under grant 140646/2000-0 for Viviane Silva. Viviane Silva and Ricardo Choren also are supported by the PRONEX Project under grant 664213/1997-9, by the ESSMA project under grant 552068/2002-0 and by the 1st art. of decree number 3.800, of 04.20.2001. The authors would also like to acknowledge the contributions of Carlos J. P. de Lucena.

References

[1] Bauer, B.: UML Class Diagrams Revisited in the Context of Agent-Based Systems. In: Wooldridge, M., Weiss, G. and Ciancarini, P. (Eds.) Agent-Oriented Software Engineering, Second International Workshop, AOSE 2001, Montreal, Canada (2001): 101-118.

[2] Caire, G.: MESSAGE: Methodology for Engineering Systems of Software Agents Initial Methodology. In: Technical report, EDIN 0224-0907, Project P907, EURESCOM (2001).

[3] Foundation of Intelligent Physical Agent: FIPA Interaction Protocols Specification, (2003). Available at URL http://www.fipa.org/repository/ips.html

[4] He, M., Jennings, N., Leung, H.: On agent-mediated electronic commerce. In: IEEE Transaction on Knowledge and Data Engineering, v.15, n.4, (2003): 985-1003.

[5] Huhns, M., Singh, M.: Agents and Multi-agent Systems: Themes, Approaches and Challenges. In: Huhns, M. Singh, M. (Eds.): Readings in Agents. Morgan Kaufmann (1998): 1–23.

[6] Jennings, N., Wooldridge, M.: Applications of Intelligent Agents. In: Jennings, J., Wooldridge, M. (Eds.), Agent Technology: Foundations, Applications, and Markets (1998): 3-28.

[7] Object Management Group. OMG Unified Modeling Language Specification, version 1.3, March 2003.
[8] Odell, J., Parunak, H., Bauer., B.: Extending UML for Agents. In: Odell, J., Parunak, H. and Bauer, B. (Eds.), Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National Conference on Artificial Intelligence (2000): 3-17.
[9] 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, and Castro, J., (Eds.), Software Engineering for Large-Scale Multi-Agent System, LNCS, Springer-Verlag, (2003).
[10] Silva V., Lucena C.: From a Conceptual Framework for Agents and Objects to a Multi-Agent System Modeling Language. In: Technical Report CS2003-03, School of Computer Science, University of Waterloo, Canada (2003). (submitted to publication)
[11] Silva, V., Lucena, C.: Extending the UML Sequence Diagram to Model the Dynamic Aspects of Multi-Agent Systems, In: Technical Report MCC15/03, PUC-Rio. Rio de Janeiro, Brazil (2003). (submitted to publication)
[12] Wagner, G.: The Agent-Object-Relationship Metamodel. In: Proceedings of the 2nd International Symposium: From Agent Theory to Agent Implementation together with EMCRS 2000, Vienna, Austria (2000).
[13] Wagner, G.: Agent-Oriented Analysis and Design of Organizational Information Systems. In: Proceedings of Fourth IEEE International Baltic Workshop on Databases and Information Systems, Vilnius, Lithuania (2000).