

Modeling Multi-Agent Systems

Viviane Torres da Silva

Carlos José Pereira de Lucena

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

PUC-RioInf.MCC 06/04 March, 2004

Abstract.

Different methodologies, languages and platforms for multi-agent systems propose very distinct and varied sets of abstraction. In this context, there is a need for creating a conceptual framework that defines the frequently used multi-agent system abstractions, their relationships and their behavior. As it is the case with any new software engineering paradigm, the successful and widespread deployment of multi-agent systems require modeling languages, among other agent-based software technologies, that explore the use of agent-related abstractions and promote the traceability from the design models to code.

This paper contemplates the definition of a multi-agent system conceptual framework called TAO and of a multi-agent system modeling language called MAS-ML. Our goals are to describe the structural and dynamic aspects of the abstractions commonly used in multi-agent systems by defining a conceptual framework, to propose a modeling language that describes structural and dynamic diagrams to model such aspects and to present the traceability from the structural models into code.

Keywords. Multi-agent system, conceptual framework, modeling language, automatic code generation, UML.

Resumo.

Diferentes metodologias, linguagens e plataformas para sistemas multi-agentes propõem abstrações variadas e com definições muito diferentes. Nesse contexto, é necessário criar frameworks conceituais que definam as abstrações, seus relacionamentos e seus comportamentos. Como em qualquer novo paradigma para engenharia de software, o sucesso e a difusão de sistemas multi-agentes requer, entre outras tecnologias de software baseadas em agentes, linguagens de modelagem que explorem o uso de abstrações relacionadas a agentes e promovam o refinamento dos modelos de design para código.

Esta paper contempla a definição de um framework conceitual para sistemas multi-agentes chamado TAO e uma linguagem de modelagem para sistemas multi-agentes chamada MAS-ML. Os objetivos desta tese são descrever os aspectos estáticos e dinâmicos das abstrações frequentemente utilizadas em sistemas multi-agentes definindo um framework conceitual, propor uma linguagem de modelagem que descreva diagramas estáticos e dinâmicos para modelar tais aspectos e descrever o refinamento dos modelos estáticos para código.

Palavras-chave. Sistema multi-agente, framework conceitual, linguagem de modelagem, geração automática de código, UML.

1. Introduction

Multi-agent systems (MASs) are gaining wide acceptance in both industry and academia as a powerful paradigm for designing and developing software systems [10]. Together with this growth, new methodologies, methods, modeling languages, development platforms, tools and programming languages are being proposed. Agent-based systems require adequate techniques that explore their benefits and their peculiar characteristics. However, different methodologies, languages and platforms for MAS propose very distinct and varied sets of abstractions. It is often very difficult to understand the definition of each abstraction and the relationships between them. In this context, there is a need for creating a *conceptual framework* that defines the abstractions, their relationships and their behavior.

As it is the case with any new software engineering paradigm, the successful and widespread deployment of MASs require *modeling languages*, among other agent-based software technologies, that explore the use of agent-related abstractions and promote the traceability from the design models to code. 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 related to the entities behavior [9].

To reduce the risk when adopting a new technology it is convenient to present it as an incremental extension of known and trusted methods, and to provide explicit engineering tools that support industry-accepted methods of technology deployment [5]. A modeling language for multi-agent system preferably should be an incremental extension of a known and trusted modeling language.

Since agents and objects coexist in MASs, the UML modeling language [5] can be used as a basis for developing MAS modeling languages. The UML modeling language is a *de facto* standard for object-oriented modeling. UML is used both in industry and academy for modeling object-oriented systems. Nevertheless, in its original form UML provides insufficient support for modeling MASs. Among other things, the UML meta-model does not provide support for modeling agents, organizations and agent roles.

2. The need for a Conceptual Framework for MASs

After exhaustive review of theories, methodologies and methods for multi-agent systems, we felt the need for a *conceptual framework* that defines the commonly used abstractions found in MASs. Few conceptual frameworks were proposed for describing MAS concepts [1,3,12]. Such frameworks do not define several structural and dynamic aspects commonly described in MASs.

2.1. The need for defining the structural aspects of MASs

Different agent-based techniques describe MASs based on different entity kinds. Each technique describes different properties and associates different relationships with each entity. Therefore, there is a need for defining the structural aspects of MASs by describing the entities frequently found in MASs. While describing the entities it is necessary to define the properties associated with them and their relationships. The relationships between the properties should also be described.

2.2. The need for defining the dynamic aspects of MASs

The dynamic aspects are characterized by the internal execution of the entities (intra-actions) and by the interactions between the entities. Different entities may execute and interact in different ways. Since MASs are composed of different entities, there is a need for describing the dynamic aspects of such entities.

The intra-actions of an entity are related to the behavioral properties that it defines. For instance, the intra-actions of objects are related to the execution of methods and the intra-actions of agents are related to the execution of actions and plans. The interactions between one entity and another are influenced by the relationships that link the entities. Although agents interact by sending and receiving messages, the sequence of messages and the content of each message sent and received by agents are influenced by their relationships. Therefore, there is a need for describing the interactions between the entities based on the relationships that link them.

3. The need for a MAS Modeling Language

Several modeling languages for MAS that extend the UML meta-model have been proposed [2,5,11]. However, there still is 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, (iii) models the structural and dynamic aspects frequently described in MASs and (iv) provides the traceability from the design models to code.

3.1. The need for representing MASs entities as first-class abstractions

A modeling language for MASs should define MASs entities as first-class abstractions. All proposed modeling languages describe *agent* as a first-class abstraction. However, entities such as *role*, *organization* and *environment* are not defined as such in many of them. Due to this limitation such languages cannot be used to model several structural and dynamic aspects of MASs. It is not possible to model the relationships and the interactions between agents, objects and other MAS entities.

3.2. The need for an explicit description of a MAS meta-model

A meta-model defines a language for specifying models by describing the semantics of a set of abstractions and by defining how such abstractions get instantiated [9]. For each abstraction, the meta-model describes its semantic, the meta-relationship with other abstractions and the graphical representation of such abstraction in models.

Several proposed modeling languages that extend UML do not clearly describe the extensions applied to the UML meta-model. Although they describe extensions to UML diagrams, such languages usually do not describe how the UML meta-model was extended in order to model new abstractions. The modeling languages describe the graphical representation of the new abstractions but do not clearly describe their semantics or the relationships between them.

The modeling languages [5,11] that describe the extensions applied to the UML meta-model use stereotypes based on the meta-class *Class* (that represents object classes) to define agents. Since agents and objects do not share the same properties and relationships agents should not be described based on objects.

3.3. The need for modeling structural aspects of MASs

A MAS modeling language should describe structural diagrams to model the structural aspects of MASs. The set of structural diagrams need to be capable of modeling (i) the entities usually defined in MASs, (ii) the properties of such entities by associating the properties with the entities, and (iii) the relationship between the entities. The modeling languages proposed in the literature do not model several MAS entities and therefore do not define the relationships between agents and these entities.

In order to model MAS entities, properties and relationships, UML structural diagrams need to be extended. Different diagram elements¹ can be created to represent MAS entities, properties and relationships. Different diagram elements facilitate the visualization and modeling of such abstractions. If the modeling language defines more than one structural diagram, each diagram should describe the set of entities, properties and relationships that can be modeled. It is also important to specify if the diagrams define different views of the same abstractions or if they model different sets of abstractions.

3.4. The need for modeling dynamic aspects of MASs

MAS dynamic diagrams should be defined by a MAS modeling language to model the dynamic aspects of such systems. The dynamic diagrams need to be capable of modeling (i) the interactions between the entities defined in the structural diagrams and (ii) the internal execution of such entities. MAS dynamic diagrams can be defined by extending UML dynamic diagrams while defining the interactions and intra-actions of MAS entity instances.

Different interaction kinds need to be modeled in the MAS dynamic diagrams. The different entities that compose MASs interact in different ways. The MAS dynamic diagrams should also model the different internal behavior of the MAS entities. Moreover, different diagram elements should be created to represent the MAS entity instances.

Several proposed modeling languages do not represent the different interactions kinds related to objects and agent-related abstractions. Moreover, many of them do not model the internal execution of the agent-related abstractions.

3.5. The need for the traceability from MAS design models to code

The development of appropriate approaches to implement agent-based systems is a key issue in getting the agent technology into the mainstream of software development. In order to implement MASs designed using a MAS modeling language, it is necessary to transform the MAS design models into code. MAS design models are high-level models that are composed of agent-related abstractions. To transform MAS models into code, agent-related abstractions need to be mapped into abstractions defined in the programming language.

4. The TAO Conceptual Framework

The TAO (Taming Agents and Objects) conceptual framework [8] goal 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

¹ Diagram elements are elements used to graphically represent abstractions in diagrams.

object-oriented methodologies, languages, and theories. TAO groups together the abstractions that are frequently described in the literature for MASs. The benefit of having such framework is to provide support for developing new methodologies, methods and languages based on the essential concepts defined and related in the framework. Each concept is viewed as candidate abstraction for modeling languages, methodologies and support environments to be applied in different phases of the MAS development.

TAO defines the structural and dynamic aspects of MASs. While describing the structural aspects of MASs, TAO defines the entities that may be described in MASs, their properties and the relationships associated with them (Table I). The dynamic aspects described in TAO are classified in primitive dynamic processes and high-level dynamic processes (

Table II). The primitive dynamic processes describe the creation and destruction of entities. High-level dynamic processes are more complex domain-independent behavior that are described based on primitive dynamic processes. The domain-independent high-level dynamic processes describe patterns of behavior derived from the characteristics of the inhabit, ownership and play relationships between the MAS entities because these relationships are domain-independent relationships. Agents, organizations and objects inhabit environments. Agents and sub-organizations play at least on role. Every role is owned by an organization.

Table I – TAO entities, properties and relationships

<i>Entities</i>	Object	Agent	Organization	Agent Role	Object Role	Environment
Object	specialization association aggregation dependency	association	association	association	association play	inhabit
Agent	association	specialization	---	play	---	inhabit
Organization	association	---	specialization	ownership play	ownership	inhabit
Agent Role	association	play	ownership play	specialization control association aggregation dependency	dependency association	---
Object Role	association play	---	ownership	dependency association	specialization association aggregation dependency	---
Environment	inhabit	inhabit	Inhabit	---	---	specialization association
<i>Properties</i>	attribute method	goal belief action plan	goal belief action plan axiom	goal belief duty right protocol	attribute method	attribute method or goal belief action plan

Table II – TAO primitive and high-level dynamic processes

	Primitive Dynamic Processes	High-Level Dynamic Processes
Object	creation destruction	---
Agent	creation destruction	entering an organization leaving an organization moving from an environment to another
Organization	creation destruction	entering an organization leaving an organization moving from an environment to another
Agent Role	creation destruction	---
Object Role	creation destruction	---
Environment	creation destruction	---

5. The MAS-ML Modeling Language

The MAS-ML (Multi-Agent System Modeling Language) [6,7] goal is to model all the structural and dynamic aspects defined in TAO. The MAS-ML meta-model is defined extending the UML meta-model according to the concepts defined in TAO. When extending UML according to TAO concepts, it is not possible to use only the tag, constraints and stereotypes extensions mechanisms provided by UML. New meta-classes and new stereotypes associated with new entities, properties and relationships defined in TAO and not presented in UML were incorporated to the UML meta-model. Since our aim is to produce a conservative extension of UML, meta-classes defined in UML were not modified during the extension. 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. Figure 2 shows the new meta-classes related to the relationships described in TAO that have been proposed by the MAS-ML to the UML meta-model.

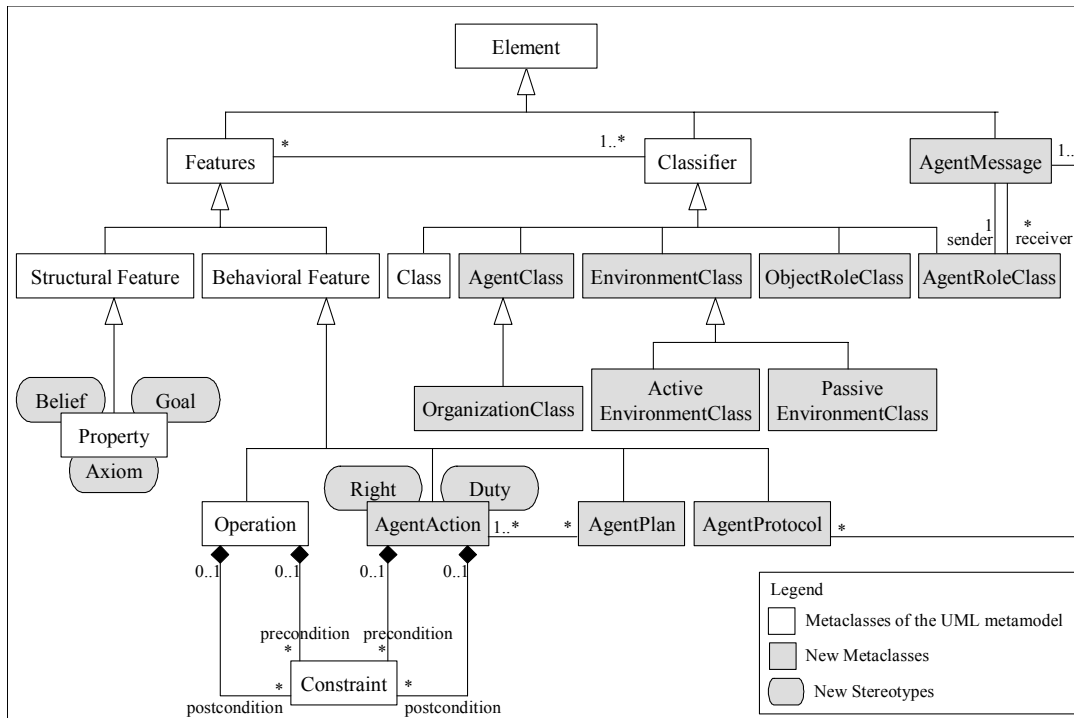


Figure 1 – The extended UML meta-model incorporating the MAS-ML entities and properties

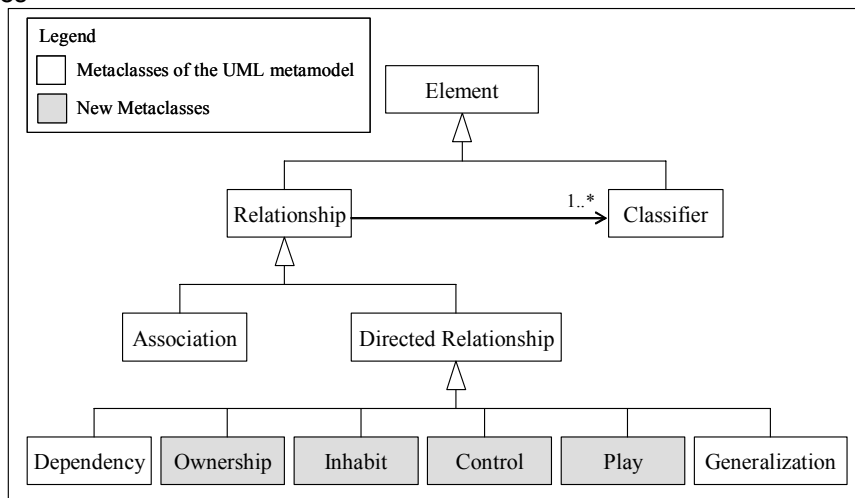


Figure 2 – The extended UML meta-model incorporating the MAS-ML relationships

5.1 Structural diagrams

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. MAS-ML extends the UML class diagram to represent the structural relationships between agents, agents and classes, organizations, organizations and classes, environments, and environments and classes. The purpose of the organization diagram is to model the system organizations and the relationships between them and other system entities. The role diagram is responsible for modeling the relationships between the roles defined in

organizations. The three structural diagrams – class, organization and role diagrams – model all entities and all relationships defined in TAO.

5.2 Dynamic diagrams

Furthermore, we propose to extend the UML sequence diagram to represent the dynamic aspects of MASs. Using the dynamic sequence diagram it is possible to model (i) the interaction between agents, organizations, environments, and objects, (ii) the execution of plans and actions associated with agents, organizations and active environments, and (iii) to model protocols defined by roles. The entities that are modeled in the sequence diagrams are instances of the entity classes modeled in the structural diagrams. The methods, plans and actions modeled in the sequence diagrams are also defined in the structural diagrams associated with the respective entity classes.

Since one sequence diagram can model different entities executing different plans and actions at the same time and can also model the same entity playing more than one role, the sequence diagram can express concurrency and parallelism in the design models. Moreover, since a sequence diagram can model different entities executing in different environments and an entity moving from an environment to another, the sequence diagram can express distribution in the design models. The representation of both the structural and dynamic diagrams together with their use for modeling concrete MAS applications can be found in [6,7].

5.3 Generating code from structural diagrams

With the aim of implementing systems modeled using MAS-ML, a transformer was developed to generate code from the MAS-ML structural diagrams. The models described at the agent level of abstraction are transformed into object-oriented code.

The transformation process is composed of three phases (Figure 3). In the first phase, the MAS-ML graphical models of the application are described textually by using the MAS-ML grammar. Using the MAS-ML grammar it is possible to describe all the information presented in MAS-ML structural diagrams. It is possible to describe the entities, their properties and relationships.

In the second phase, domain-independent entities rules and domain-dependent entities rules are applied to the textual description of the MAS-ML models generating a partial transformation. The domain-independent entities rules generate the set of domain-independent object-oriented modules and their relationships defined by an object-oriented abstract architecture. The set of classes that compose the architecture represents the entities that cannot be directly mapped from MAS-ML into Java classes. Entities such as agents, organizations and agent roles cannot be directly mapped into classes because classes are defined based on attributes and methods and these entities are defined based on properties such as goals, actions and protocols. The domain-dependent entities rules generate classes that represent the application entities. Such classes extend the abstract classes defined in the architecture.

The third phase applies the domain-dependent relationships rules to the output of the second phase. The classes of the previous phase are modified in order to represent the application relationships. The output of this phase is a set of OO Java classes that represent the application modeled using MAS-ML.

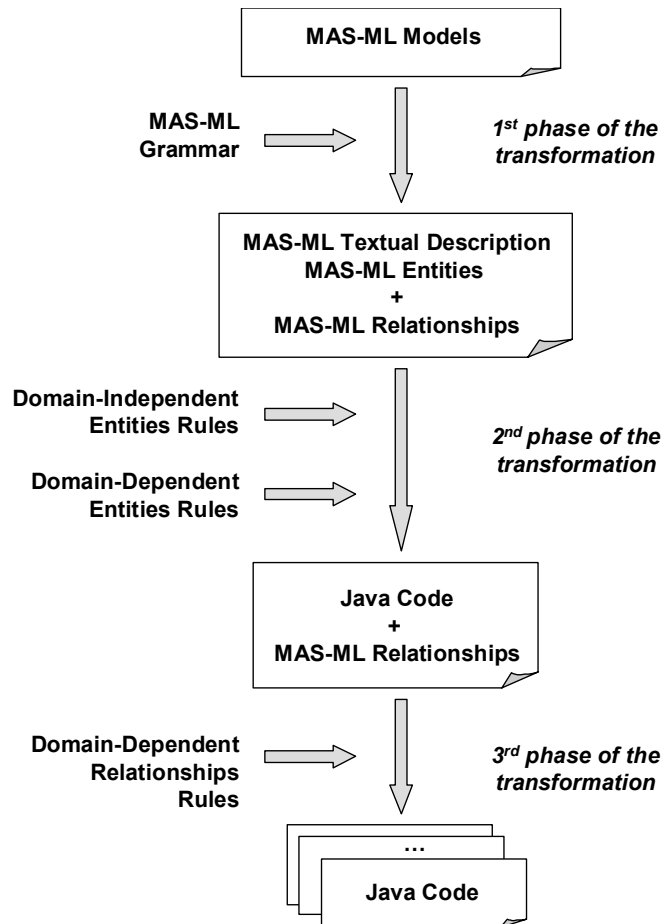


Figure 3 – The transformation phases

6. The Relationship Among UML, TAO and MAS-ML

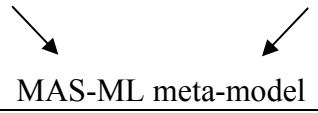
To better explain the relationship among UML, TAO and MAS-ML we use a four-layer metadata architecture described in the MOF specification [4]. The four architecture layers are: meta-meta-model layer, meta-model layer, domain model layer and instance layer (Table III).

The meta-meta-model layer comprises the description of the structure and the semantics of meta-metadata. The meta-meta-model layer specified by OMG is MOF. TAO uses the ER model (Entity-Relationship model) to describe the entity and relationship meta-metadata that appear in this layer.

The meta-model layer comprises the description of the structure and semantics of metadata. OMG defines the UML meta-model that is an instance of the MOF meta-meta-model. We define the TAO meta-model (or conceptual framework) that is an instance of the ER meta-meta-model. The MAS-ML meta-model extends the UML meta-model according to the concepts described in the TAO. MAS-ML specifies a modeling language that incorporates the object and agent-oriented concepts.

The domain model layer depicts the data specific to the application domain. The concepts modeled using MAS-ML are instantiated according to the domain information creating the domain models. The instance (information or implementation) layer describes the specific instances of the domain model data.

Table III – MOF meta-data architecture

Layers	Models	
Meta-meta-model layer	MOF meta-meta-model	ER meta-meta-model
Meta-model layer	UML meta-model	TAO meta-model
	 MAS-ML meta-model	
Domain model layer	MAS-ML models	
Instance layer	Instances of the domain models	

7. Conclusion

Different methodologies, languages and platforms for MAS propose very distinct and varied sets of abstraction. The main role of the TAO framework is to provide a conceptual framework to understand distinct abstractions, their relationships and interactions in order to support the development of large-scale MASs. The proposed framework elicits an ontology that connects consolidated abstractions, such as objects and classes, and frequently used MASs abstractions (agents, roles, organizations and environments), which are the foundations for agent and object-based software engineering.

In order to define a MAS modeling language that contemplates all the concepts described in TAO, we propose the MAS-ML language. MAS-ML extends UML by preserving all object-related concepts that constitute the UML meta-model while including the agent-related concepts described in TAO. Using MAS-ML it is possible to describe agents, roles, organizations and environments, to model the interactions among them and to model the internal execution of such entities.

References

- [1] DARDENNE, A.; LAMSWEERDE, A.; FICKAS, S. Goal-directed Requirements Acquisition. *Science of Computer Programming*. v.20, p.3-50. 1993.
- [2] DEPKE, R.; HECKEL, R.; HUSTER, J. M. Formal agent-oriented modeling with UML and graph transformation. In: *Science of Computer Programming archive: Special issue on applications of graph transformations (GRATRA 2000)*, Netherlands: Elsevier, v.44, n.2, p. 229-252. 2002.
- [3] D'INVERNO, M.; LUCK, M. *Understanding Agent Systems*. New York: Springer. 2001.
- [4] MOF: Meta Object Facility Specification, version 1.4, OMG. Available at: <<http://www.omg.org/cwm>>. Accessed in: February 14th 2004.
- [5] ODELL, J.; PARUNAK, H.; BAUER, B. Extending UML for Agents. In: WAGNER, G.; LESPERANCE, Y.; YU, E. (Eds.), *In: Proceedings of the Agent-Oriented Information Systems Workshop*, Austin, p. 3-17. 2000.
- [6] SILVA, V.; CHOREN, R.; LUCENA, C. Using the MAS-ML to Model a Multi-Agent System. In: LUCENA, C.; GARCIA, A.; ROMANOVSKY, A.; CASTRO J.; ALENCAR, P. (Eds.) *Software Engineering for Large-Scale Multi-Agent Systems II*, LNCS 2940, Springer, 2004. (to be published)

- [7] SILVA, V.; LUCENA, C. From a Conceptual Framework for Agents and Objects to a Multi-Agent System Modeling Language, In: SYCARA, K.; WOOLDRIDGE, M. (Edts.), *Journal of Autonomous Agents and Multi-Agent Systems*, Kluwer Academic Publishers, 2004. (to be published in March).
- [8] 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*. LNCS 2603, Berlin: Springer, 2003.
- [9] UML: Unified Modeling Language Specification, version 2.0, OMG, Available at: <http://www.omg.org/uml/>. Accessed in: February 14th 2004.
- [10] WOOLDRIDGE, M.; CIANCARINI, P. Agent-Oriented Software Engineering: the State of the Art. In: CIANCARINI, P.; WOOLDRIDGE, M. (Eds.) *Agent-Oriented Software Engineering*, LNCS 1957, Berlin: Springer, p. 1-28. 2001.
- [11] WAGNER, G. The Agent-Object-Relationship Metamodel: Towards a Unified View of State and Behavior. In: *Information Systems*, v.28, n.5, 2003.
- [12] 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.