ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal
Regular Issue, Vol. 8 N. 4 (2019), 49-81
eISSN: 2255-2863
DOI: http://dx.doi.org/10.14201/ADCAIJ2019844981

NorMAS-ML: Supporting the Modeling of Normative Multi-agent Systems

Emmanuel Sávio Silva Freire[a], Mariela Inés Cortés[b], Robert Marinho da Rocha Júnior[b], Enyo José Tavares Gonçalves[c], and Gustavo Augusto Campos de Lima[b]

[a]Teaching Department, Federal Institute of Ceará, Morada Nova, Brazil
[b]Computer Science Department, State University of Ceará, Fortaleza, Brazil
[c]Computer Science Department, Federal University of Ceará, Quixadá, Brazil
savio.freire@ifce.edu.br, mariela.cortes@uece.br, robster@gmail.com, enyo@ufc.br, gustavo@larces.uece.br

KEYWORD

Normative Multi-agent System; Norms; Modeling Language; NorMAS-ML

ABSTRACT

Context: *A normative multi-agent system (NMAS) is composed of agents that their behavior is regulated by norms. The modeling of those elements (agents and norms) together at design time can be a good way for a complete understanding of their structure and behavior. Multi-agent system modeling language (MAS-ML) supports the representation of NMAS entities, but the support for concepts related to norms is somewhat limited. MAS-ML is founded in taming agents and objects (TAO) framework and has a support tool called the MAS-ML tool.* ***Goal***: *The present work aims to present a UML-based modeling language called normative multi-agent system (NorMAS-ML) able to model the MAS main entities along with the static normative elements.* ***Method***: *We extend the TAO adding normative concepts and spread out those concepts in two syntaxes of MAS-ML. Either abstract, adding or updating metaclasses and stereotypes or concrete, defining new graphic elements for representing the elements defined in the abstract syntax. Besides, we evolve the MAS-ML tool, considering the extension of MAS-ML by the model-driven approach.* ***Results***: *NorMAS-ML, the new version of MAS-ML, allows a complete view of NMAS entities and has a support tool called NorMAS-ML tool. Beyond the definition of NorMAS-ML and its tool, we generate a new static diagram called "norm diagram" supported by the NorMAS-ML tool. In order to illustrate the syntax of NorMAS-ML, the entities of a conference management system and its norms are modeled jointly.* ***Conclusion***: *NorMAS-ML can help software designers (i) to understand the properties and behavior of NMAS entities and (ii) to provide a software modeling following the stakeholders' need and less complex for the development phase.*

1. Introduction

In the context of complex and highly dynamic environments, software engineers and developers search for mechanisms that allow their systems detect changes in the environment where they are inserted and plan the most appropriated action (Russell and Norvig, 2003). Agent-centered systems (Ferber *et al.*, 2004) have been widely explored by the scientific community as a suitable approach for the development of complexity systems (Luck and D'Inverno, 1995). However, the development of these systems is not trivial, requiring an effort of Software Engineering to provide appropriate support to various development activities.

A software development process comprises well-founded phases, and each phase is composed of a set of specified activities, methods and tools to achieve higher product quality and productivity (Sommerville, 2006). In order to reach the quality level and the user and designer's expectations, agent-oriented software engineering (AOSE) (Jennings and Wooldridge, 2000) investigates the characteristic and behavior of autonomous agents (Jennings, 1996), providing methodologies, modeling, and programming languages for multi-agent system (MAS). The system is composed of agents sharing information, cooperating or disputing between each other (Russell and Norvig, 2003).

In normative multi-agent systems (NMAS) (Boella *et al*., 2006), agents are placed in environments and they communicate with other agents in order to reach their goals, and the agent's behavior can be regulated by a set of norms. Researches in NMAS are focused on regulating the behavior of the agents by describing the actions that can be performed or states that can be achieved (permissions), actions that must be performed or states that must be achieved (obligations), and actions that cannot be performed or states that cannot be achieved (prohibitions) (Figueiredo and da Silva, 2011).

A norm is composed of the following static elements: deontic concept, involved entities, actions, activation constraints, sanctions, and context (Figueiredo and da Silva, 2011). Besides, norms can be defined at design or run time (López y López, 2003). The modeling of norms at design time allows a complete vision of the system and can influence the modeling of other system's entities. Some verification and checks can still be done in norm-definition phase and the faults in the system could be detected and corrected before the system implementation phase (López y López, 2003).

Due to importance of the joint modeling of MAS main entities and norms, modeling NMAS projects by modeling languages is required. In this scenario, a few numbers of modeling languages allow the modeling of the NMAS entities. These include multi-agent system modeling language (MAS-ML) (da Silva and de Lucena, 2003) and normative modeling language (NormML) (Figueiredo and da Silva, 2011). Although, MAS-ML allows the modeling of the all MAS main entities based on taming agent and objects (TAO) framework (Silva *et al*., 2003), the support for modeling norms is somewhat limited. In contrast, NormML allows the modeling of the static elements norms and has mechanisms to solve norm conflict at design time. However, it does not support the modeling of structural and behavioral aspects of entities, such as agent, agent role, object role, object, organization and environment. Thus, the effectiveness of NormML depends on its integration with other modeling language in order to represent all NMAS entities.

The main goal of this paper is to present a UML-based modeling language called normative multi-agent system (NorMAS-ML) able to model the MAS main entities along with the static normative elements. To accomplish our goal, we have extended MAS-ML and its conceptual framework TAO, while considering the normative concepts and structures defined in NormML. Furthermore, the modeling tool of MAS-ML has been extended following the new structures defined in NorMAS-ML. We have chosen MAS-ML since (i) it is based on UML, (ii) it has a suitable ontology to design main MAS entities and their relationships, (iii) it has partially support to model some normative elements, and (iv) it has a modeling tool. In order to demonstrate the NorMAS-ML syntax, a modeling example was performed on conference management context system (Zambonelli *et al*., 2001; Dignum, 2004; Harmon *et al*., 2008; Figueiredo, 2011).

We presented a preliminary version of our proposal in a previous work (Freire *et al*., 2012). However, we did not present in (Freire *et al*., 2012) the metamodel, constraints, templates, modelling tool and presented a summarized version of the illustration.

The paper is structured as follows. We describe the base notion related to agent and multi-agent systems, TAO, MAS-ML and NormML in Section 2. Section 3 presents related work involving modeling languages and organizational models. The definition of NorMAS-ML is described in Section 4. Section 5 presents the definition of NorMAS-ML tool. In Section 6, we modeled a Conference Management System, aiming to illustrate the contribution of this work. Finally, conclusions and future work are discussed in Section 7.

2. Background

This section describes the main concepts that support this work, including TAO conceptual framework and MAS-ML and NormML modeling languages.

2.1. TAO: Taming Agents and Objects

TAO framework was defined by Silva *et al*. (2003) (Silva *et al*., 2003) as an ontology which has the Software Engineering fundamental based on agents and objects. Consequently, it supports the MAS development on a large scale. The static and dynamic aspects of TAO are described in following subsections.

2.1.1. *Static Aspects of TAO*

TAO has the following defined entities: agent, object, organization, role (agent role and object role) and environment (Silva *et al*., 2003). Each one of them has owner properties (state and behavior) and relationships with other entities. A state defines information about other system's entities while a behavior defines the actions (operations) that an entity can execute. A relationship can link an entity with others and describes how they are related to each other. Figure 1 shows the entities defined in TAO metamodel. Each one of them is described below.

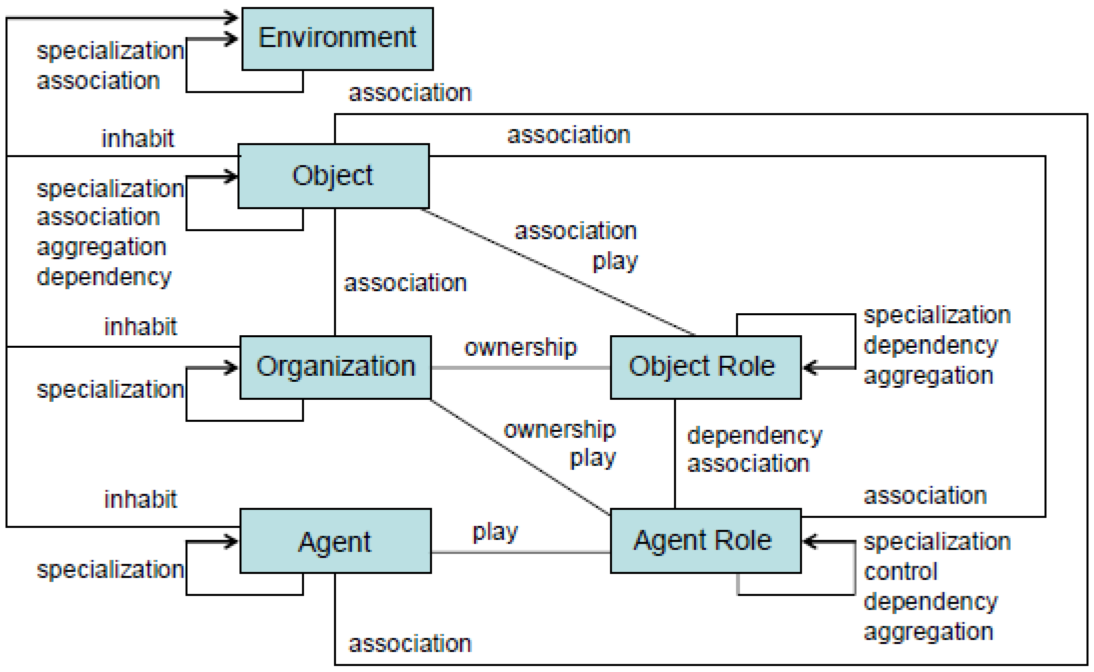


Figure 1: Relationships and entities of TAO (Silva et al., 2003).

- Object: it is a passive element which has a state and a behavior. During owner life cycle, an object executes operations that can change owner previous state, but these operations can be executed when them are requested by other system's entities;
- Agent: it is an autonomous, adaptable, and interactive element which has the following mental components: beliefs (represent all information that an agent knows), goals (correspond to future states that an agent want to achieve), plans (are formed by actions, organized in a sequence, that represent a goal would be reached) and actions;

- Organization: it is responsible to put together agents, objects, and suborganizations. An organization has goals, beliefs (as agents), and axioms and it is placed in an environment. Axioms are global restrictions of an organization and agents and suborganizations should be followed these restrictions. An organization defines agent roles which should be executed by agents and suborganizations and object roles which should be executed by objects;
- Object role: it guides and restricts the object's behavior to limit owner information and owner behavior that other system's entities can access. An object role can add information, behavior, and relation to object's instance which executes this role;
- Agent role: it guides and restricts the agent's behavior describing owner goals and defining the actions that can or must be performed by an agent while it plays an agent role. An agent role defines duties, rights, and protocols. A duty states an action that must be performed by an agent; a right defines an action that can be performed by an agent; and a protocol defines the interaction between an agent role and others system's entities;
- Environment: it is an element which agents, objects, and organizations are placed in.

Additionally, the following relationships are defined in TAO:

- Inhabit: it specifies that a citizen (the entity's instance that inhabits) is created and destroyed at habit and can entry and leave of that habit, following its permissions. A citizen cannot reside in two habitats in same time. A habitat knows all citizens which reside in it and each citizen knows its habitat. This relationship is applied in environments and agents, environments and objects, and environments and organizations;
- Ownership: some elements need to be members of other elements. The relationship ownership specifies that an element – a member – is defined in the scope of other element – an owner – and that a member need to follow a set of global restrictions defined by an owner. Members can be created and destroyed dynamically by its owner;
- Play: objects, agents, and suborganizations need to be related to roles. The relationship play defines what objects, agents, or suborganizations are related to a role and must assume the properties and the relationships defined by this role. These entities' behavior is directed and restricted by this role's scope;
- Specialization/Generalization: this relationship defines that a sub-element, when specializes a super element, can add and redefine properties and behavior related to the super-element;
- Control: it defines that a controlled entity must do all that the controller asks. The controller knows own controlled entities and each controlled entity its controller. This relationship can be used between two agent roles, but it cannot use between two object roles;
- Dependency: an element – client – can be set to depend on another entity – supplier – to execute its work. This relationship defines that a client cannot do completely own work without the supplier's support;
- Association: if an element is associated with another, it knows that this other element exists. This relationship should define how an element interacts with other;
- Aggregation/Composition: if an element is aggregated with other, it means that this element is part of an aggregator. An aggregator can use the functionalities available in own parts. Parts do not need to know that they are being aggregated by an aggregator, but an aggregator knows each one of own parts.

2.1.2. *Dynamic Aspects of TAO*

TAO also describes the internal interaction and execution of owner entities, highlighting the behavioral domain-independent process. This process can be divided into two groups. Either primitive or high-level process. Primitive processes are composed of creation and destruction processes while high-level process are made of: (i) process of commitment with a role, (ii) process of role activation, (iii) process of cancellation of commitment with a role, (iv) process of role deactivation, and (v) process of dislocate from environment to other. Besides, dynamic aspects describe the behavioral patterns derivative of characteristics of inhabit, ownership and, play relationships between MAS entities. Other process for agents entering or leaving an organization, organizations entering or leaving an organization, and agents or organizations entering or leaving an organization.

2.2. MAS-ML: Multi-Agent System Modeling Language

MAS-ML is a modeling language that extends UML (OMG, 2018) to allow the modeling of MAS based on TAO conceptual framework (Silva *et al*., 2003). The extension process was based on the definition of new metaclasses and stereotypes in order to represent the MAS entities. From TAO's ontology, MAS-ML gives support to development MAS of large scale. Figure 2 presents MAS-ML metamodel (partial).

2.2.1. *Static Aspects of MAS-ML*

Static diagrams defined in MAS-ML are detailed as follows.

- Class Diagram: MAS-ML extends the UML class diagram in order to represent the relationships between classes and other MAS entities. The extended class diagram performs the relationships between classes and environments, classes and agents, and classes and organizations. Moreover, it also was extended to represent the relationships between agents, environments, and organizations;
- Organization Diagram: it allows to model all organizations of a system. The organization diagram is responsible to model the properties of an organization (goals, beliefs, plans, actions and axioms), the roles defined by an organization, the entities (agents, classes and suborganizations) that play these roles and the environment that it habits (da Silva *et al*., 2005). The relationships ownership, play and inhabit can be used in this diagram;
- Role Diagram: it is responsible to show the relationships between agent role and object roles identified in organization diagrams. This diagram also identifies the classes accessed by object and agent roles. The interactions between agents and organizations are described via relationships between roles defined in role diagrams (Silva *et al*., 2008). The relationships control, dependency, association, aggregation, and specialization can be used in this diagram.

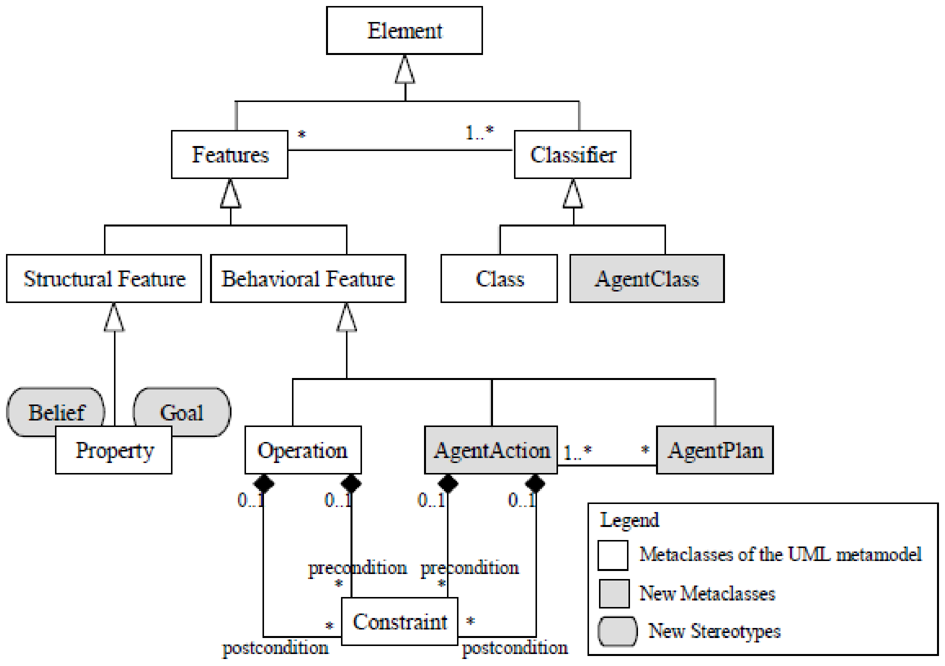


Figure 2: MAS-ML's metamodel and its properties (da Silva et al., 2005).

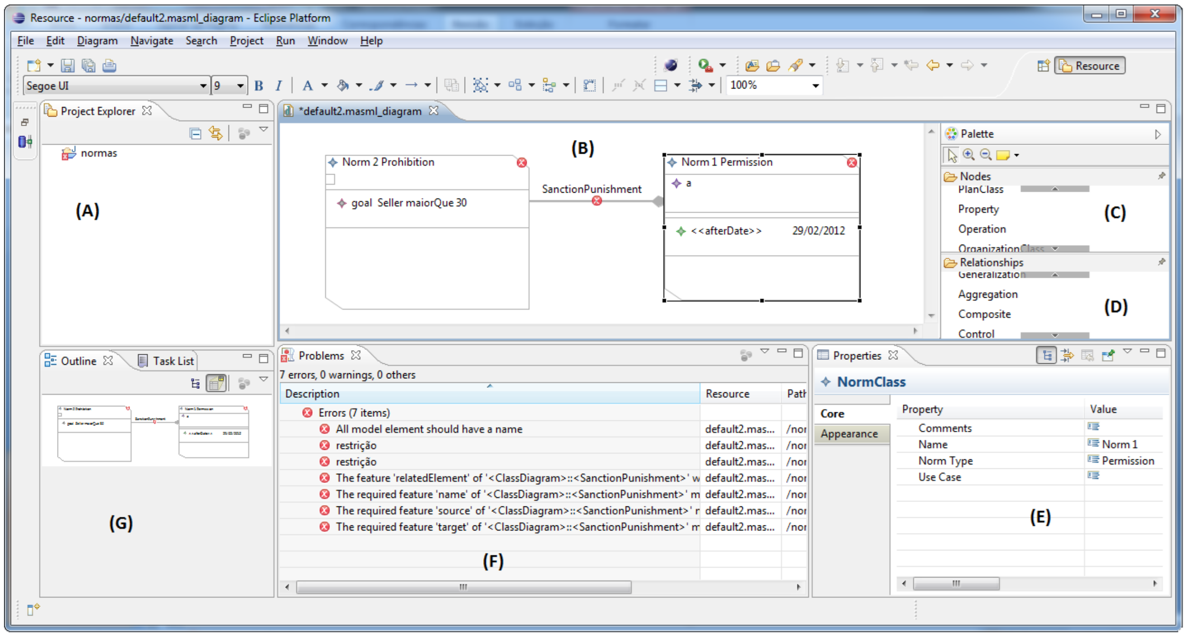
2.2.2. *Dynamic Aspects of MAS-ML*

Dynamic aspects of MAS-ML are represented by means of an extension of UML sequence and activity diagrams in order to represent the interactions between MAS instances and the actions of each instance. The extension of sequence diagram (Silva *et al*., 2003) includes the definition of new pathnames and icons for MAS instance (agents, organizations, and environment). The concept of message used in UML was extended in order to represent entities that are sending and receiving messages and they do not call methods of other entities. Furthermore, stereotypes were defined in order to represent the creation and the destruction of MAS instances, and to represent the interaction between agents, organizations, objects and own roles. Some stereotypes associated with messages were redefined and others were created by Silva (Silva *et al*., 2003). From adaptation in MAS-ML, in activity diagram, it is possible to model plans and actions of agents and organizations, and the concepts related to the modeling language (da Silva *et al*., 2005). Because of this, each activity is represented by a rectangle with round borders. The agent's beliefs are represented by a square jointly the identification of beliefs used by an agent and the goals are represented at right superior corner through an textual description with << *goal* >> stereotype.

2.2.3. *MAS-ML tool*

MAS-ML tool is a modeling environment to model MAS entities (Silva *et al*., 2003). Through this tool, software developers can work with problem domain and can use the concepts defined in solution domain simultaneously. These concepts are related to agent paradigm. This tool allows the modeling of class, organization, role and sequence diagrams defined in MAS-ML tool. This tool was development as a plug-in of eclipse platform (Eclipse, 2018). So, users can model MAS and use the resources offered by this platform. Several implementation languages (Braubach *et al*., 2003) (Bellifemine *et al*., 2007; Bordini *et al*., 2007) have been used Java in order to develop agent platform and then the use of Eclipse can get easy a possible code generation in the same implementation environment.

MAS-ML tool was development following the model-driven approach considering MAS-ML metamodel as a central model. Figure 3 shows a general view of MAS-ML tool and its components. The letters in this figure represent the following resources:



7 errors, 0 warnings, 0 others

Description	Resource	Path
Errors (7 items)		
All model element should have a name	default2.mas...	/nor
restrição	default2.mas...	/nor
restrição	default2.mas...	/nor
The feature 'relatedElement' of '<ClassDiagram>::<SanctionPunishment>' w	default2.mas...	/nor
The required feature 'name' of '<ClassDiagram>::<SanctionPunishment>' m	default2.mas...	/nor
The required feature 'source' of '<ClassDiagram>::<SanctionPunishment>' r	default2.mas...	/nor
The required feature 'target' of '<ClassDiagram>::<SanctionPunishment>' m	default2.mas...	/nor

NormClass

Core | Appearance

Property	Value
Comments	
Name	Norm 1
Norm Type	Permission
Use Case	

Figure 3: MAS-ML tool and its resources.

- Package Explorer (A): it allows the organization of files in a tree-like structure in order to get better the management and manipulation of files;
- Modeling View (B): it gives support to the visualization and edition of model in iterative form;
- Nodes Palette (C): the constructors used in diagram stay in nodes palette. Therefore, the users can create instances of these constructors;
- Relationship Palette (D): it shows the relationships can be used in diagrams;
- Properties View (E): it allows the setting of model properties;
- Problems View (F): it shows the model inconsistencies;
- Outline View (G): it presents the distribution of created model.

2.3. NormML: Normative Modeling Language

NormML (Figueiredo and da Silva, 2011) is UML-based modeling language to allow the behavior restriction of MAS entity. NormML metamodel is based on SecureUML metamodel (Basin *et al*., 2006) providing a modeling language of functions, permissions, actions, resources and restriction of authorization taken together relationships between permissions and roles, actions and permissions, resources and actions, and restrictions and permissions. NormML includes a set of invariants that guarantees a well-formed of norm and various operations can be used to identify conflicts between two defined norms. Therefore, this modeling language allows the norm modeling to restrict the behavior of agents, organizations and suborganizations during a period of time, and defines sanctions can be applied when a norm is violated or followed (da Silva Figueiredo *et al*., 2011). The main normative elements are detailed as follows.

- Deontic concepts: deontic logic refers the logic of requests, commands, rules, laws, moral principles and judgments (Meyer and Wieringa, 1993). In NMAS, such concepts are used to describe the constraints for agent behavior by describing obligations (actions that must be performed or states that must be achieved), permissions (the actions that can be performed or states that can be achieved) and prohibitions (actions that cannot be performed or states that cannot be achieved). Thus, the most important property of a norm is the identification of deontic concept related to a norm;
- Involved entities: as norms are defined to restrict the behavior of entities, the identification of these entities is essential. A norm can regulate the behavior of individual (a determinate agent or an agent when plays a role) or a group of individuals (all agents when play a determinate role, group of agents, groups of agents playing a role or all agents stay in the system);
- Actions: since a norm is defined to restrict the entities' execution, it is important to specify the involved action. An action can be either communication, represented by sending and receiving of a message, or non-communication, such as to access and modify a resource, to enter in an organization, to move to another environment, etc;
- Activation Constraints: norms have a period during while they are active, i.e., during while their restrictions must be fulfilled. Norms can be activated by one constraint or a set of constraints that can be: the execution of actions, the specification of time intervals (before, after, between), the achievement of systems states or temporal aspects (such as dates), and also the fulfillment / violation of another norm;
- Sanctions: when a norm is violated, the entity that has violated this norm may suffer a punishment. In the same way, when a norm is fulfilled the entity who has followed the norm may receive a reward. Such rewards and punishments are called sanctions and should be described together with the norm specification;
- Context: norms are usually defined in a given context that determines the area of their application. A norm can, for instance, be described in the context of a given environment and should be fulfilled only by the agents executing in the environment. A norm can also be defined in the context of an organization and must be fulfilled only by the agents playing roles in the organization.

A norm can restrict the behavior of agents, all agents playing a role or a determinate agent playing a role, defined by agent and role relationship. Figure 4 shows NormML metamodel that a norm corresponds a set of metaclasses instanced along with their relationships. A norm can be a permission norm (should instance a NormPermission metaclass), a prohibition norm (should instance a NormProhibition metaclass) or an obligation norm (should instance a NormObligation metaclass).

NormML considers a resource with any properties of MAS entities. This language has a four kind of resources: attribute, method, entity and association. It extends the set of resources with agents' actions and roles represented by AgentAction metaclass. Therefore, it is possible to describe norms controlling the access for attributes, methods, objects and associations, and the execution of agent and roles' actions. Each kind of resource is related to a set of actions that can be used to control the resource access. For instance, the attributes are related to actions of read, update and complete access (read and update). Regarding constraints applied to action of agent and roles (AgentAction metaclass), the behavior that must be used is the action execution (ActionExecute). It is important to notice that AgentAction is a resource and ActionExecute is an action that is using to control or restrict the access to resource.

Additionally, NormML allows the specification of period of time during a norm is active. This is represented by NormConstrain metaclass. So, whether a norm is related to a Before clause, it means a norm is active only before the execution of action(s) described in Before clause. Analogously for After clause. For Between clause, a norm is only active during the period defined by two action groups. NormML tool supports the modeling of the elements described in this section and enable the verification of conflicts between norms.

3. Related Work

Some approaches using norms for agents and MAS have been proposed by (Dennis *et al*., 2010; van der Vecht *et al*., 2009; de Vries *et al*., 2009; García-Camino *et al*., 2006; Vasconcelos *et al*., 2007). These authors have investigated on the use of norms in order to restrict the entities' behavior, the modeling of elements of norms, the verification of conflict between norms and the implementation of norms. However, this section presents the analyze of research papers about organizational models and modeling languages for NMAS.

3.1. Organizational Models

Multi-Agent systems based on quadrants (MASQ) (Ferber *et al*., 2009) is a meta-model that has various aspects related to organization-centered MAS. It is based on a the 4-quadrant model by (Wilber, 1997) following the axes: internal (mental states, world representations)/external (behavior, objects and organizations), and individual/collective. Although MASQ allows the representation of permission, prohibition and obligation norms and their activation constraints in context of an organization, MASQ does not allow the definition of norms for an individual agents, agent role for all system, the modelling of norms in context of an environment, and the definition of sanctions (reward).

Moise+ (Hübner *et al*., 2002) is based on Moise (Hannoun, 2002) that presents a vision centered in an organization. Moise+ has two main notions: (i) organizational specification and organizational entity. For an organization, (Hübner *et al*., 2002) defines structural, functional and deontic aspects. Consequently, this model only allows the permission and prohibition norms for agent roles in context of an organization. These kinds of norms can restrict non-communicative actions. Moise+ has a tool called Moise API and Platform (Hübner *et al*., 2002) that allows to specify an organization and its entities. However, this organization model (i) does not allow the specification of environments, curbing the modelling of agents moving between environments, (ii) does not give support to define agent properties, and (iii) does not allow the specification of norms, sanctions neither environments.

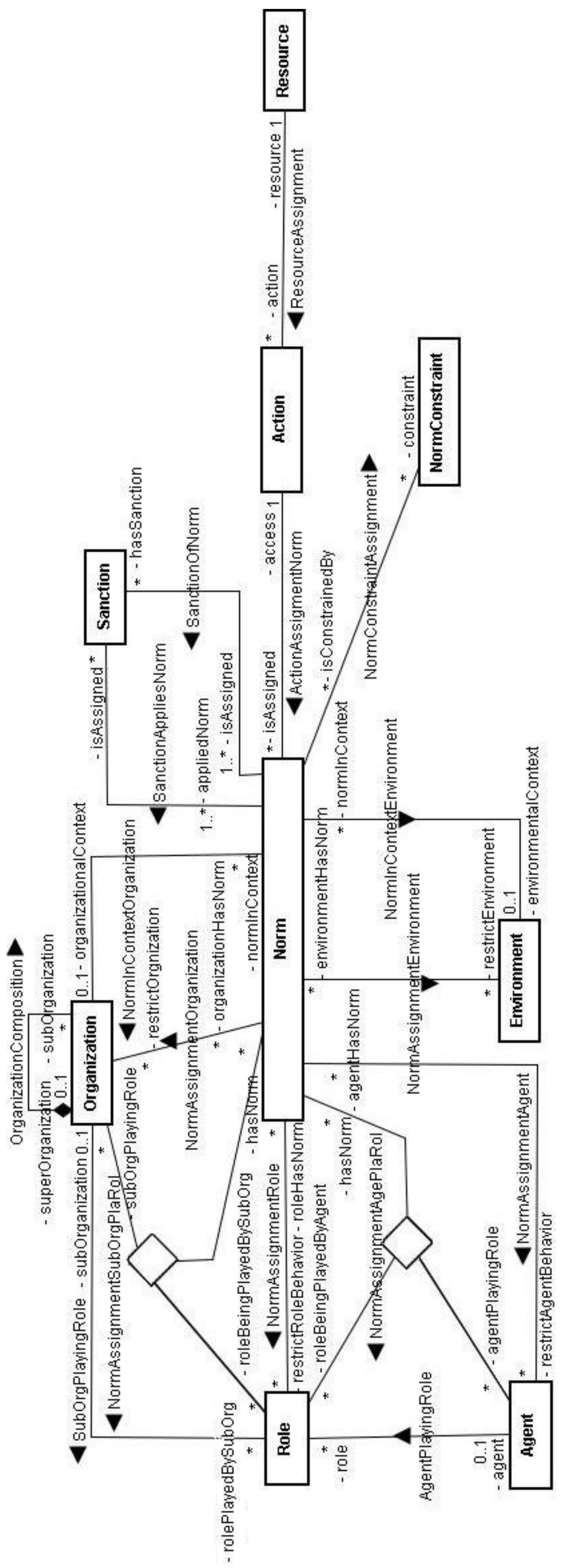


Figure 4: *NormML Metamodel (Figueiredo and da Silva, 2011).*

Dignum (Dignum, 2004) proposes OperA framework that allows the specification of MAS through the difference between the characteristics (structure and behavior) of the organizational model and the behavior of agents restrained in this model. This framework has the organizational, social, and interaction models in order to design organizations and their components. These models allow the definition of roles along with its goals, duties and obligations, and norms that regulate an organization. OperA gives support to describe obligation, permission, and prohibition norms for agents, agent roles, and groups of agents in the context of an organization. It is possible the definition of activation constraints for a norm. However, this organizational model (i) does not the modeling of structural aspects of agents, (ii) does not allow the specification of environments, so it is not possible to model agents moving between environments, (iii) does not give support to definition of norms for agents playing a role, (iv) does not offer support to definition of sanctions (reward and punishment), and (v) does not have a tool.

3.2. Modeling Languages

Several modeling languages have been proposed for MAS. However, they provide limited support to elements of norms and their relationship with MAS entities. Amongst them, AUML (Odell *et al*., 2000) and ANote (Choren and Lucena, 2005) do not support the modeling of norms. In this section, agent-oriented rule markup language (AORML), agent modeling language (AML), and Multi-agent system modelling language 2.0 (MAS-ML 2.0) were analyzed considering the supporting of modeling normative static elements and MAS entities jointly.

AORML (Wagner, 2003) is a modeling language for MAS based on agent-oriented rule (AOR) metamodel (Wagner, 2003). The AOR entities are agent, event, action, claim, committed and object. These entities are represented by stereotypes in metaclass *Class* of UML. This language allows the definition of norms for a determinate role and for a group of agents, the definition of activation constraint for norms and the restriction for non-communicative actions. However, AORML (i) does not allow the specification of environments, so it is not possible to model agents moving between environments, (ii) does not allow the modeling of agents moving between organizations, (iii) does not support the definition of norms for a group of agents, (iv) does not allow the modeling of sanctions, and (v) does not have a tool.

AML (Danc, 2008) is a semi-formal modeling language to specify, model, and document MAS. This language is based on the metamodel that extends UML without to introduce new concepts in UML metamodel. Its entities are agents, resources, and environments. AML allows to describe norms for a determinate role and for a group of individuals, to restrict communicative and non-communicative actions, and to define the period when a norm will be active. However, AML has same drawbacks: (i) it does not have totally support for dynamic of MAS, (ii) it does not offer supporting of sanctions, (iii) it does not allow the definition of norms for environment, (iv) it does not forecast the restriction of activation due to following or violation of norms, and (v) it does not have a supporting tool.

MAS-ML 2.0 (Gonçalves *et al*., 2015) is an extension of MAS-ML that allows the modeling of different agent architectures defined by (Russell and Norvig, 2003). The extension includes new metaclasses and stereotypes to represent the characteristic of each one agent architecture (Freire *et al*., 2013). The new constructs are represented in the concrete syntax by new textual stereotypes and a new modeling tool was proposed (Gonçalves *et al*., 2011). A code generation approach was proposed to this extension by (Lopes *et al*., 2018). Due the extension does not change the supporting of normative elements, MAS-ML 2.0 as with MAS-ML allows the modeling of deontic concepts of permission and obligation through << *duty* >> and << *right* >> stereotypes defines in agent role and the definition of obligation norms in context of an organization through << *axiom* >> stereotype. Nevertheless, this language (i) does not give support the definition of norms applied in an environment, (ii) does not allow the modeling of prohibition norms for agent roles, (iii) does not supporting the definition of permission and prohibition norms for agents and organizations, and (iv) does not allow the modeling of sanctions (reward and punishment) neither activation of norms.

4. Normative Multi-Agent Modeling Language

As MAS-ML is based on UML, we followed the extension process defined on UML to define NorMAS-ML. This process requires adjustments in all levels of a modeling language, such as abstract and concrete syntaxes.

In abstract syntax, new metaclasses, stereotypes or tags can be added in previous abstract syntax. New graphical elements are required in concrete syntax. In this section, we detailed this process. Initially, we extended TAO framework to include normative concepts in this framework. Further, the definition of the NorMAS-ML abstract syntax is done following the extension process of UML. Also following this process, we defined the NorMAS-ML concrete syntax.

4.1. NorMAS-ML Metamodel

This section depicts the extension of TAO framework in order to represent the normative static elements (Figueiredo, 2011). As a result, Norm concept was defined as an entity because it has a state, behavioral properties and determinate relationships. Besides, the other concepts were defined as relationships, such as: (i) Context, it is responsible to identify the context were a norm is applied, (ii) Restrict, it identifies what entities will be restricted by a norm, (iii) SanctionReward, it depicts the rewards of a norm, and (iv) SanctionPunishment, it is responsible to identify the punishments of a norm. Figure 5 shows the extension of TAO framework. In the next subsection, we detailed those elements representing them via templates.

4.1.1. *Norm Entity*

A norm is an element that restricts, during a period of time, the behavior of agents, organizations, and suborganizations, and applies sanctions when this norm is followed or violated (Figueiredo and da Silva, 2011). A Norm entity has a state that stores a resource will be restricted, and behavioral properties and relationships. (da Silva Figueiredo *et al*., 2011) defines that a resource can be an agent, an agent role, an organization and a property of an entity. The entity can be regulated by a norm can be an agent, an agent role, an organization and an environment. The properties can be governed by norms can be a goal, a belief, an attribute, a method, an action, a plan, a protocol, an association or a message.

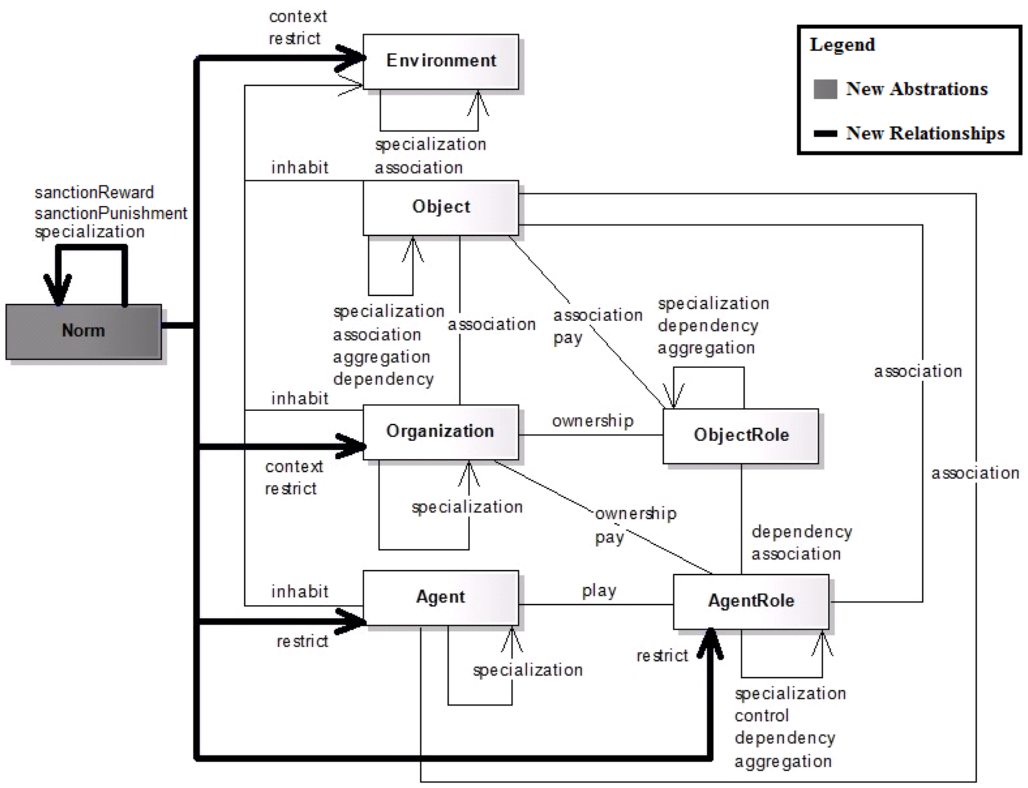


Figure 5: TAO extended metamodel.

Norm behavior is defined in its characteristics that is based on deontic concepts and on activation constraints. Deontic concepts define the kind of restriction of a norm. Thus, a norm can be an obligation (what the agent must execute), a permission (what the agent can execute) or a prohibition (what the agent cannot execute). Activation constraints define the period that a norm will be active. A norm can be activated by a restriction or a set of restrictions. A restriction can be an action execution, specified periods of time (before, after or between), the reach of system states or temporal aspects (as date), the activation or deactivation of other norms and the following or violation of a norm.

Norm relationships describe the context where a norm can be applied, the entity that has its behavior restricted by a norm and the reward or the punishment will be received by entity followed or violated a norm. Norm template shows the Norm class with its state, behavioral properties, and relationships.

Norm

Norm_Class Norm_Class_Name
Restriction_Type Deontic_Concept_Name
Resource < *Element_Class_Name.property* >
Activation_Constraint setOf {Constraint_Type Constraint_Type_Name :
(< Element_Class_Name_First > and/or < Element_Class_Name_Second >) or < date > or
< Element_Class_Name.property : Operator = (Element_Class_Name.property)or value) >}
RelationshipssetOfRelationship_Name
end Norm_Class

4.1.2. *Norm Relationships*

In this section, we present the four relationships defined in TAO in order to associate TAO entities with Norm element. Let us consider that A is a set of agents, $a \in A$, **E** is a set of environments, $e \in E$, **N** is a set of norms, $n \in N$, and **O** is a set of objects, $o \in O$. Let us consider that **Org** is a set of organizations, *org*, *subOrg* $\in$ *Org*, and **subOrg** only represents a suborganization. Let **R** be a set of roles, $R = RObj \cup RAg$, where **RObj** is a set of object roles and **RAg** is a set of agent roles, $r \in R$, $ro \in Robj$ and $ra \in RAg$. For each norm relationship will be presented its definition, its classification and the elements that can be related to it.

- *Context (C): C(context, norm): C(e, n), C(org, n), C(subOrg, n)*. When an entity class is related to a norm class by context relationship it means that entity instance is a context that determines the application area of a norm. A norm instance can be associated at least with an instance of organization, suborganization, or environment. In consequence, all inhabits of organizations, suborganizations, and agents will be their behavior regulated by that norm instance.
- *Restrict (R): R(element, norm): R(a, n), R(e, n), R(org, n), R(subOrg, n), R(ra, n). Restrict* relationship defines the entity will be restricted by a norm. When an entity class is related to a norm class by this relationship it means that entity instance is regulating by a norm and this entity can be receive a sanction whether violate or follow this norm. The entities can be regulated by a norm are agent, environment, organization, suborganization, and agent role.
- *SanctionReward (SR): SR(reward, norm): SR(n, n). SanctionReward* relationship specifies a reward that an entity that followed a norm can receive. This relationship can only relate to norm classes. Because of this, this relationship identifies the norm has a reward and the resource of other norm that will be the reward.
- *SanctionPunishment (SP): SP(punishment, norm): SP(n, n). SanctionPunishment* relationship specifies a punishment that an entity that violated a norm can receive. This relationship can only relate to norm classes. Because of this, this relationship identifies the norm has a punishment and the resource of other norm that will be the punishment.

The following template represents the new relationships defined in TAO.

```
_______________________________________Relationship_____________________________________

  Relationship Relationship_Name
  Context : context, norm
  | Restrict : element, norm
  | Sanction_Reward : reward, norm
  | Sanction_Punishment : punishment, norm
  end Relationship
________________________________________________________________________________________
```

4.2. Adjustments in TAO Entities

Besides to define *Norm* element and its relationships, some adaptations in TAO entities were required in order to preserve the correctness and consistence of the metamodel. (Silva *et al.*, 2003) defined that an agent role guides and regulates the behavior of an agent because goals, beliefs, duties, rights, protocols and commitments associated with agent role characterizes the agent while plays this agent role.

In TAO, right and duty concepts are used to define the actions that must or can be executed by agents. Considering these concepts are used to regulate the agent's behavior playing a role, they can be considered semantically equivalent to deontic concepts of permission and obligation defined by norms. Thus, right and duty concepts were replaced by norm concept in agent role template and a list of action was defined in agent role template. The list is a way to choose an action that will be restricted by a norm.

```
_______________________________________Agent_Role_______________________________________

  Agent_Role_Class Agent_Role_Class_Name
  Goals setOf{Goal_Name}
  Beliefs setOf{Belief_Name}
  Actions setOf{Action_Name}
  Norms setOf{Norm_Name}
  Protocols setOf{Interaction_Class_Name} [ setOf{Rule_Name}
  Commitments setOf{Action_Name}
  Relationships setOf{Relationship_Name}
  end Agent_Role_Class
________________________________________________________________________________________
```

An organization in TAO defines a set of rules and laws in order to regulate agents and suborganizations associated with that organization. Besides, the norm concept includes the characteristics of rules and laws (Meyer and Wieringa, 1993). Thus, similarly as agent role, we replace these concepts by norms through a set of action included in organization template.

```
_______________________________________Organization_____________________________________

  Organization_Class Organization_Class_Name
  Norms setOf{Norm_Name}
  Actions setOf{Action_Name}
  Relationships setOf{Relationship_Name}
  end Organization_Class
________________________________________________________________________________________
```

In TAO, an environment has constraints of access associated with their services and resources (Silva *et al*., 2003). These constraints are looked at norm. Consequently, we removed service and resource concepts and substituted them by norms in environment template.

```
Environment

    Environment_Class Environment_Class_Name
    Norms setOf{Norm_Name}
    Behavior setOf{Properties}
    Relationships setOf{Relationship_Name}
    Events generated: setOf{Event_Name}, perceived: setOf{Event_Name}
    end Environment_Class
```

4.3. NorMAS-ML Abstract Syntax

In this section, we present the extension of MAS-ML abstract syntax in order to allow the modeling of MAS-ML entities along with norm elements in consistency with the TAO extension defined in previous section. Our strategy for extension is based on UML mechanisms that allows definition and update of metaclasses and stereotypes.

4.3.1. *Norm*

Norms are based on deontic concepts that allow the description of behavior constraints for agents, such as obligation, permission, and prohibition. Since a norm has a state, behavioral properties, and relationships, we defined *Norm* metaclass as a specialization of *Element* metaclass from the UML metamodel. For each kind of norm was defined with a stereotype: << *obligation* >>, << *permission* >>, and << *prohibition* >>. In this case, stereotypes are sufficient to distinguish these kinds since all of them have the same structure. Figure 6 shows the *Norm* metaclass stereotypes.

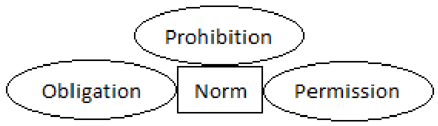


Figure 6: Norm metaclass stereotypes.

Taking into account the evolution on TAO metamodel, the MAS-ML stereotypes << *duty* >> and << *right* >> related to *AgentAction* metaclass no longer needed since norms of obligation, permission or prohibition in NorMAS-ML can be associated with agent roles to regulate the behavior of agents and suborganizations. Similarly, the concept of axioms used to characterize global constraints in an organization was replaced by norms. Thus, the stereotype related to *Property* metaclass was removed. Instead that, an organization or suborganization states actions that can, must or cannot be executed by agents or suborganizations.

(Figueiredo, 2011) states that a norm can be associated with an organization, an agent, an agent role and an environment. Considering that these entities are specializations of *Classifier* metaclass in MAS-ML metamodel, a new relationship with *Norm* metaclass was required. Some specializations of *Classifier* have no association with *Norm*, such as *Class* and *ObjectRoleClass* metaclass. This restriction was done through object constraint language (OCL) rules. Figure 7 shows the relationships defined between *Norm* metaclass and specializations of *Classifier* metaclass.

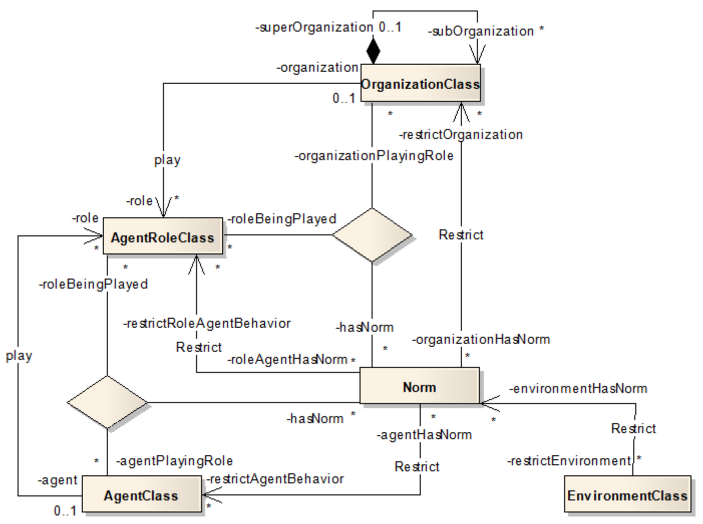


Figure 7: Relationships between Norm metaclass and AgentClass, AgentRoleClass, EnviromentClass, and OrganizationClass.

4.3.2. *Action*

Originally, the *NormAction* metaclass in MAS-ML is used in order to represent the actions about resources, that will be restricted by norm. By *Resource* relationship, a resource can be associated with an action. Through *ActionAssignmentNorm* relationship (See Figure 8), an action can be associated with one or more norms. Thus, it is possible to model norms that define different access constraints for different resources.

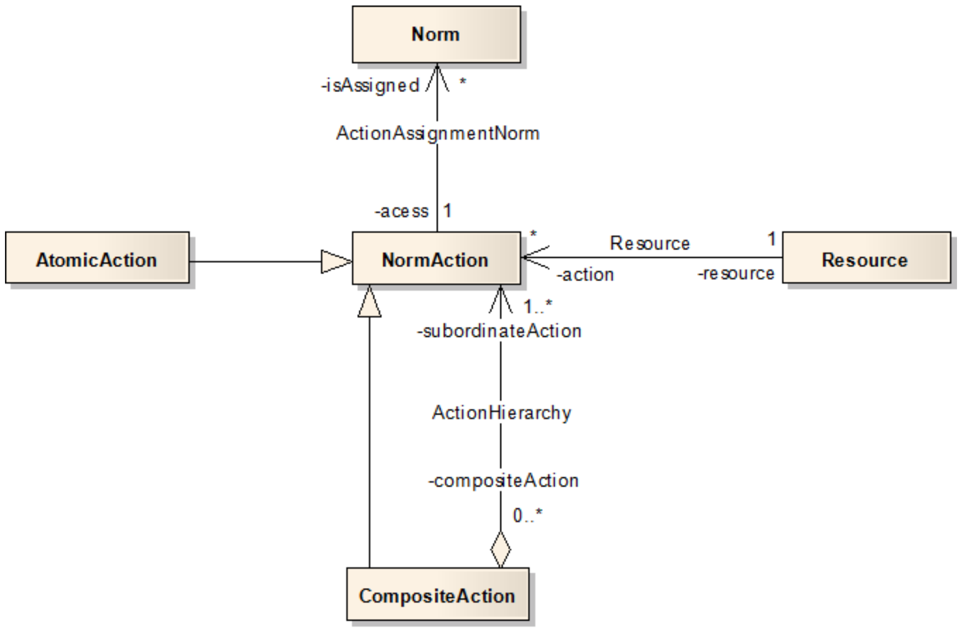


Figure 8: Relationships between NormAction and other entities.

We defined *NormAction* metaclass in NorMAS-ML metamodel. This metaclass is defined as a specialization of *BehavioralFeature* UML metaclass, and it has two specializations: *AtomicAction* metaclass and

CompositeAction metaclass. Atomic actions can be classified in actions of update, send, delete, read, receive, cancel, create, execute, achieve and commit (Basin *et al*., 2006). On the other hand, composite actions have the following kinds: update, read, full access, send, receive and execute considering determinate resources of system. To represent these concepts, we defined new stereotypes in NorMAS-ML metamodel related to *AtomicAction* and *CompositeAction* metaclasses (See Figure 9) in order to distinguish them semantically, since they had the same structure. In addition, the hierarchy of *AtomicAction* and *CompositeAction* metaclasses (Basin *et al*., 2006) is represented through of the *ActionHierarchy* relationship.

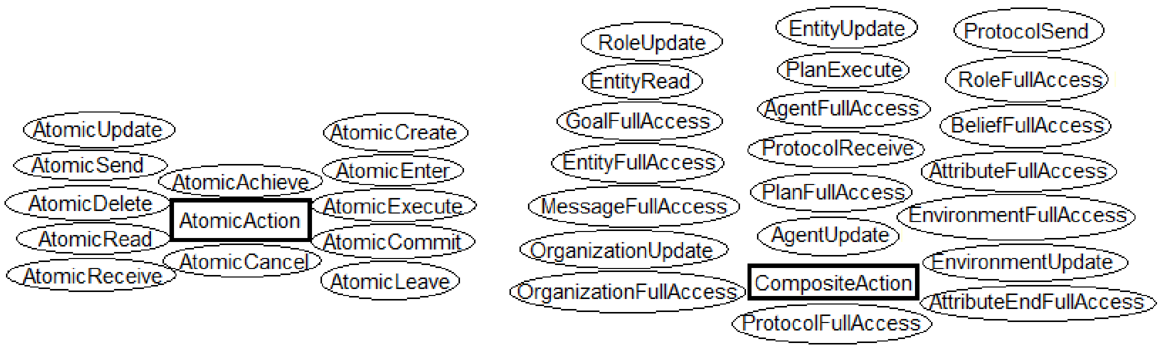


Figure 9: Stereotypes of AtomicAction and CompositeAction metaclasses.

4.3.3. *Activation Constraints*

Norms can be activated by a constraint or a set of constraints (Figueiredo, 2011), such as the execution of actions, a period of time (before, after, and between them), reaching system's states or temporal aspects (as dates), activation or deactivation of other norm, and following or violating of a norm. Upon activation, norms remain active for a period of time.

In order to modeling these concepts, we defined the *NormConstraint* metaclass. This metaclass allows to define a constraint period of a norm and has the following specializations: (i) *Before*, it indicates that a norm is active before an action execution or a determinate date; (ii) *After*, it indicates that a norm is active after an action execution or a determinate date; (iii) *Between*, it allows that a norm is active between an execution of a couple of actions or a couple of dates; (iv) *If*, it actives a norm following the comparison of two operands or dates. An operand can be a goal, a belief, an attribute or a value.

Through *NormAction* metaclass and its relationships, we can define the period of activation constraint of a norm. *BeforeAction* relationship means that a norm is active before the execution of an action, while *BeforeDate* and *After* allowing to describe that a norm is active before or after a determinate date, respectively.

To represent a norm that is active between the execution of two actions or two dates, we defined the following relationships: *BetweenBeforeAction*, *BetweenAfterAction*, *BeforeBetweenDate* and *AfterBetweenDate*. Also, the definition of constraint base on the comparison of two operands or dates is done by *CondicionalConstraint* relationship. We reused *Date* metaclass defined in NormML in order to store dates used in activation constraints.

By *condicionalConstraint* relationship, a norm will be activated when a date defined in *If* clause is achieved or operands defined in *condicionalOperand* relationship and the operator defined in *restrictOperand* relationship is true. For instance, a norm will be active whether a goal will be active is equals to the goal defined in *If* clause. A operand can be a goal, a belief, an attribute or a value. We used *Operator* metaclass to represent the comparison will be done in *If* clause. Figure 10 presents the relationships of *NormConstraint* metaclass and its specializations.

4.3.4. *Resource*

The access to resource of the system can be controlled by a set of action defined in norm context. A resource can be an attribute, a method, an association, an agent, a role, an organization, an environment, an action, a message, a protocol, a belief, a goal, or a plan (Figueiredo, 2011). In this context, *Resource* metaclass (defined in SecureUML and NormML) was incorporated in the NorMAS-ML metamodel as a specialization of *Element* metaclass to model resources.

Despite (Figueiredo and da Silva, 2011) suggested an inheritance relationship between *Resource* metaclass and MAS entities, we defined this relationship in the metamodel as an association between *Classifier*, *Feature*, *Association* and *AgentMessage* metaclass. We did this project decision in order to conserve the UML metaclasses without semantic loss. Consequently, *Resource* metaclass can be related to (see Figure 11):

- a structural (goals, beliefs, and attributes) or behavioral (methods, actions, plans, and protocols) characteristic by means of an association between *Resource* and *Feature* metaclasses;
- a classifier by means of an association between *Resource* and *Classifier* metaclasses. Thus, an agent, an organization, an agent role, and an environment can be defined as resources in the system;
- a relationship by means of an association between *Resource* and Association metaclasses allowing the regulate of access to read and update;
- a message by means of an association between *Resource* and *AgentMessage* metaclasses.

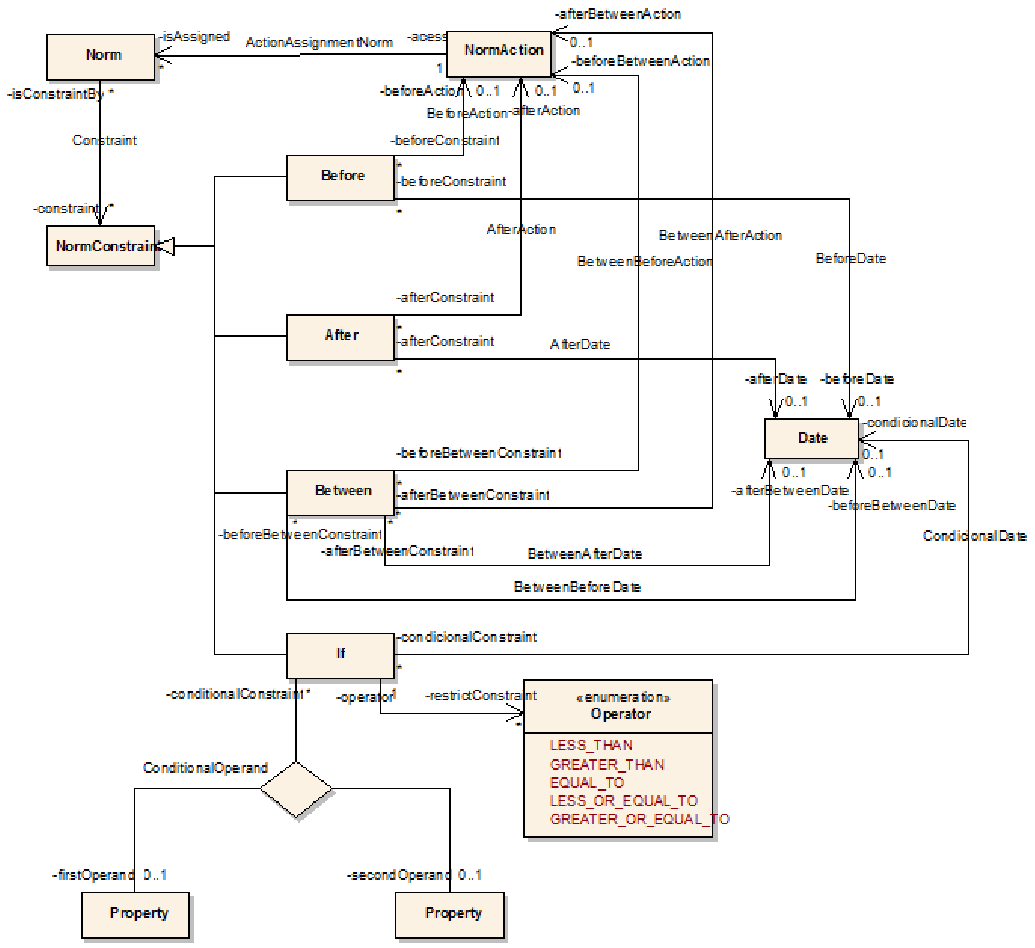


Figure 10: NormConstraint metaclass and its relationships and its specializations.

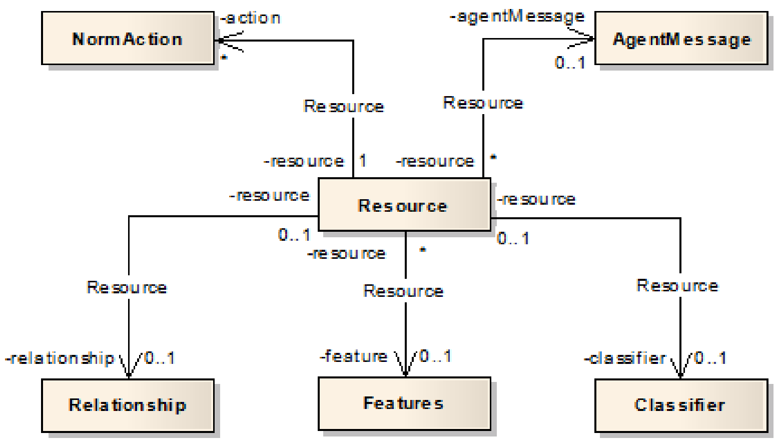


Figure 11: Associations between Resource metaclass and MAS-ML entities.

4.3.5. *Sanction Relationship*

The entity that followed or violated a norm may be subject to a sanction (Figueiredo and da Silva, 2011). We defined this concept in NorMAS-ML adding the *Sanction* metaclass as a specialization of *DirectedRelationship* metaclass. Although NormML metamodel defined punishment and reward sanctions as metaclasses, we defined them using the stereotypes << *punishment* >> and << *reward* >> because they are the same structure. Figure 12 shows these stereotypes.

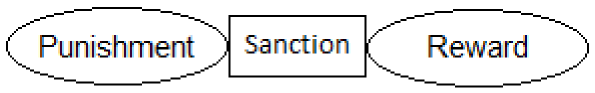


Figure 12: Stereotypes of Sanction metaclass.

4.3.6. *Context Relationship*

A norm is defined regarding a context where all entities that inhabit this context will be regulated by this norm. Thus, we defined the *Context* metaclass as a specialization of *DirectedRelationship* metaclass in NorMAS-ML metamodel. This relationship can associate a *Norm* metaclass with *Organization*, *Suborganization* or *Environment* metaclass.

4.3.7. *Restrict Relationship*

We defined *Restrict* metaclass as a specialization of *DirectedRelationship* metaclass that represents the *Restrict* relationship defined in TAO. This relationship indicates the entity that will be regulated by a norm and it can be used between a Norm metaclass and *Agent*, *AgentRole*, *Organization*, *Suborganization* or *Environment* metaclasses.

Figure 13 represents the NorMAS-ML metaclasses, regarding the original MAS-ML metamodel and our extensions related to the concepts of norms defined by (Figueiredo and da Silva, 2011) and TAO extension showed in previous section.

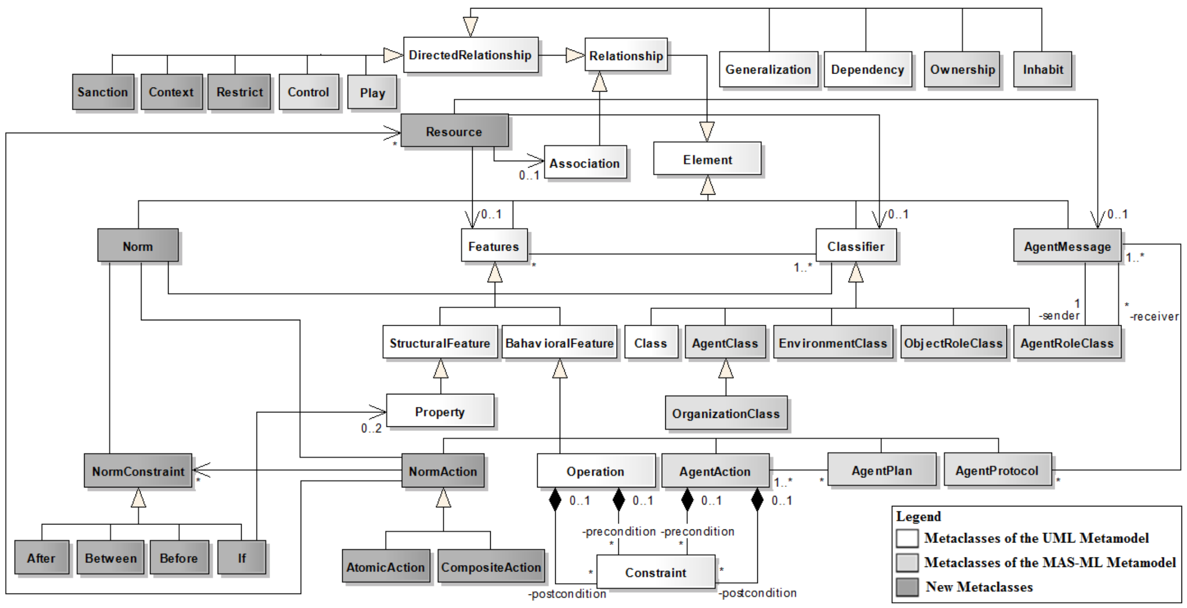


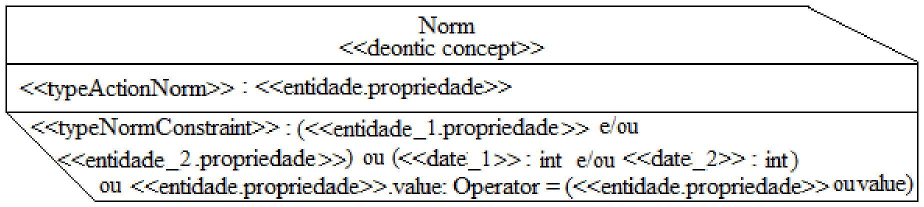
Figure 13: NorMAS-ML metamodel.

4.4. NorMAS-ML Concrete Syntax

New graphic elements in the concrete syntax was created in order to represent the new metaclasses and stereotypes defined in NorMAS-ML abstract syntax. A norm is represented as a solid rectangle with an angle in the upper right corner and another in the lower left corner (See Figure 14). The superior compartment in the figure includes the name of Norm class that it must be unique for a set of norms. Moreover, it needs to inform the deontic concept related to this norm using one of the following stereotypes: << *permission* >>, << *obligation* >>, or << *prohibition* >>.

In the intermediate compartment, we can register the resource that will be regulated by a norm. This resource can be an entity (agent, agent role, organization or environment) or a property (goal, belief, attribute, method, action, plan, protocol, association or message). The restriction type is defined by means of stereotypes from the *NormAction* metaclass (See Figure 8).

In the last compartment, we can detail a set of activation constraints of a norm. In order to define a constraint, we need to inform the kind of activation constraint via stereotypes defined in *NormConstraint* metaclass.



Norm
<<deontic concept>>
<<typeActionNorm>> : <<entidade.propriedade>>
<<typeNormConstraint>> : (<<entidade_1.propriedade>> e/ou
<<entidade_2.propriedade>>) ou (<<date_1>> : int e/ou <<date_2>> : int)
ou <<entidade.propriedade>>.value: Operator = (<<entidade.propriedade>> ou value)

Figure 14: Graphic element of Norm

Context relationship is represented as simple line with an inverted triangle in its end point (See Figure 15). This inverted triangle indicates the context (*Organization class or an Environment class*) of a norm.

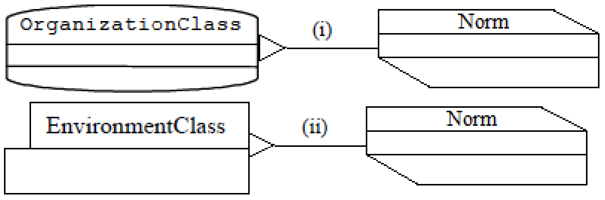


Figure 15: Graphic element of Context relationship

Restrict relationship associated with a *Norm* Class is defined as simple line with a filled square in its end point (See Figure 16). This filled square indicates the entity restricted by a norm, such as an *AgentClass*, an *OrganizationClass*, an *AgenteRoleClass* or an *EnvironmentClass*.

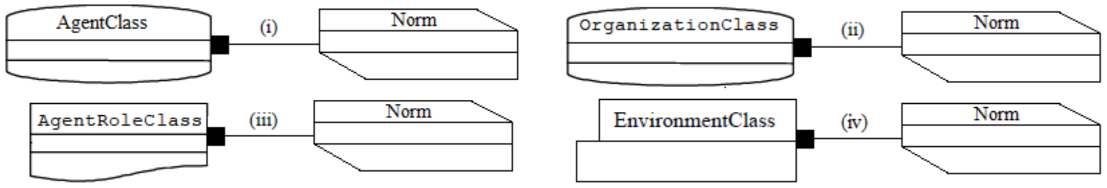


Figure 16: Graphic element of Restrict relationship

Sanction relationship is represented as simple line with a pentagon in its end point that indicates the sanction of a norm. Whether the pentagon is filled, it means that the sanction is a punishment. Otherwise, the sanction is a reward. Figure 17 shows this relationship.

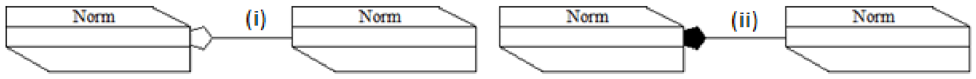


Figure 17: Graphic element of Sanction relationship.

Adjustments in graphic elements was required in consistency with the changes in their structures. Since the elimination of axiom concept in the abstract syntax, the list of axioms in the intermediate compartment in the *Organization* class no longer required. This concept was substituted by norms though Context relationship. Figure 18 shows the new graphic element of *Organization Class*. The other compartments remained unchanged.

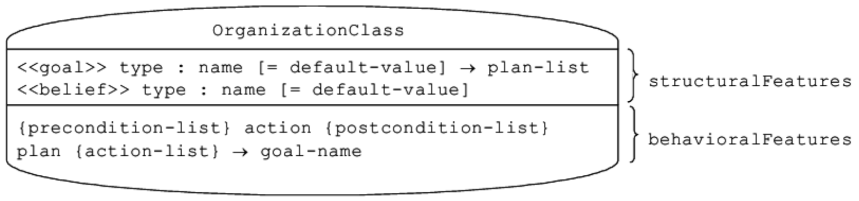

OrganizationClass
<<goal>> type : name [= default-value] → plan-list
<<belief>> type : name [= default-value]
{precondition-list} action {postcondition-list}
plan {action-list} → goal-name
structuralFeatures
behavioralFeatures

Figure 18: New graphic element of Organization Class

Due to elimination of duty and right concepts related to *agent role* metaclass, duty and right lists in the *AgentRole* class was replaced by the list of actions that can be regulated by a norm. The other compartments remained unchanged. Figure 19 presents the new graphic element of *AgentRole* Class.

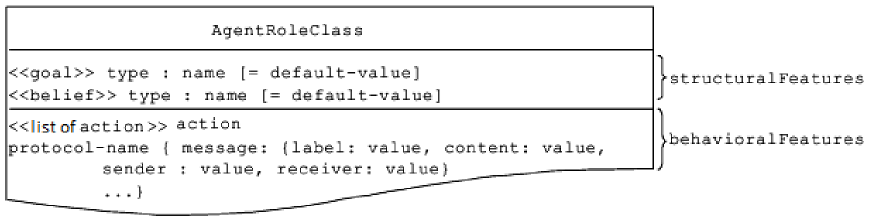

AgentRoleClass
<<goal>> type : name [= default-value]
<<belief>> type : name [= default-value]
structuralFeatures
<<list of action>> action
protocol-name { message: (label: value, content: value, sender : value, receiver: value) ...}
behavioralFeatures

Figure 19: New graphic element of AgentRole Class.

Although *EnvironmentClass* metaclass had been changed, its graphical element remains unchanged because it represents resources and services as attributes and methods. Consequently, we can represent these attributes and methods associated with norms in order to regulate the access them by other entities.

NorMAS-ML comprises a new structural diagram called *Norm Diagram* that allows the modeling of normative multi-agent systems and its properties. The entities can be represented in the *Norm Diagram* are *class*, *norm class*, *object role class*, *agent class*, *agent role class*, *organization class* and *environment class*. The following relationships can be used in this diagram are:

- *Ownership*: it is used between *organization* class and *agent role* class;
- *Play*: it is used between *agent* class and *agent role* class, between *suborganization* class and *agent role* class, and between *class* and *object role* class;
- *Inhabit*: it can used between *environment* class and *organization*, *agent role*, object role, *object*, *agent* or *norm* class;
- *Context*: it is used between *norm* class and *environment* or *organization* class;
- *Restrict*: it can used between *norm* class and *environment*, *organization*, *agent role* or agent class.
- *Sanction*: it is used between *norm* classes.

Figures 29, 31, 30, and 32 show examples of Norm Diagram.

5. NorMAS-ML Modeling Tool

This section presents the extension of MAS-ML tool in order to allow the modeling of NorMAS-ML entities and the *Norm diagram* defined in previous section. The version of MAS-ML tool used in our extension (Farias *et al*., 2009) supports the modeling of *class*, *organization* and *role diagrams* defined in MAS-ML. We changed the structure of *Organization* and *Role* diagrams and defined *Norm diagram* in order to follow the concepts and graph elements defined in NorMAS-ML. The new version of the tool is called NorMAS-ML tool. The evolution process used to create NorMAS-ML tool is detailed as follows.

Firstly, we extended the **domain model** of MAS-ML tool using the essential meta-object facility (EMOF). We included in this metamodel the *NormClass*, *ConstraintClass*, *DateConstraint*, *CondicionalConstraint*, *NormResource*, *Context*, *SanctionPunishment*, *SanctionReward* and *Restrict* metaclasses and semantics in order to represent the stereotypes defined in abstract syntax of NorMAS-ML. After this, we extended the **graph model** that define the entities, their properties and relationships, Due to this extension, we added new entities and relationships following to NorMAS-ML. Figure 20 shows our extension.

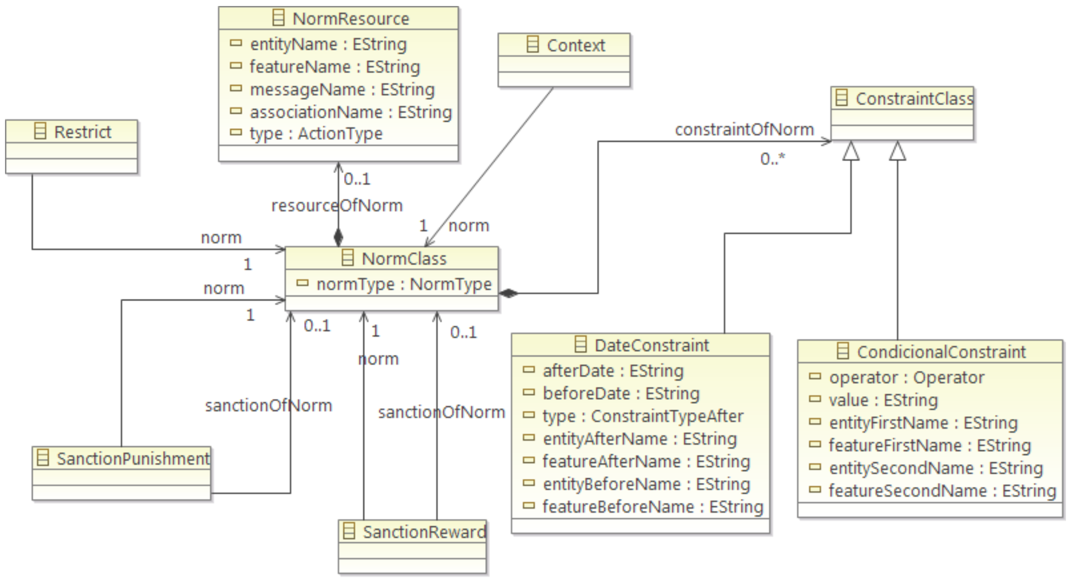


Figure 20: New metaclasses and their relationships in MAS-ML tool

Besides, we extended the **tool model** in order to include the new elements that will compose the tool palette in order to create norm and its relationships, considering the domain and **graph models** defined previously. After this, we extended the definition of graph model to represent the elements in a diagram and then, we defined compartments in order to represent a norm and its relationships.

On next, we create the **mapping model** by the combination of all steps showed previously. This mapping is used as input in process to create the specific model of platform. We used OCL in order to specify well-formedness rules to check the model consistency. Finally, we used the generative approach (Czarnecki and Eisenecker, 2000) to generate the tool considering the model defined in previous step. Figure 21 presents the dashboard defined in Eclipse where it is possible to identify all steps detailed in this section.

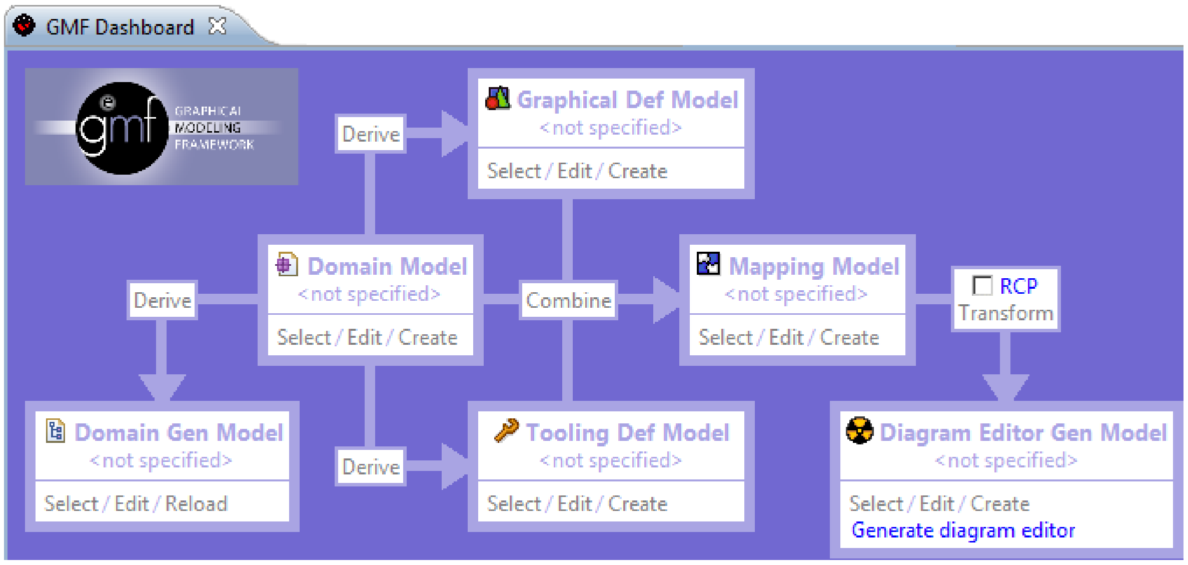


Figure 21: Dashboard of Eclipse to generate the NorMAS-ML tool.

Figure 22 shows the new elements defined in MAS-ML tool: *Norm Class* (A), *Context relationship* (B), *Restrict relationship* (C), *SanctionReward relationship* (D) and SanctionPunishment relationship (E). Also, the new representation of *OrganizationClass* (F) and *AgentRoleClass* (G) is showed in the same figure. We used OCL rules in order to validate the models created in NorMAS-ML tool that can be access by the validate option in the edit menu (See Figure 23).

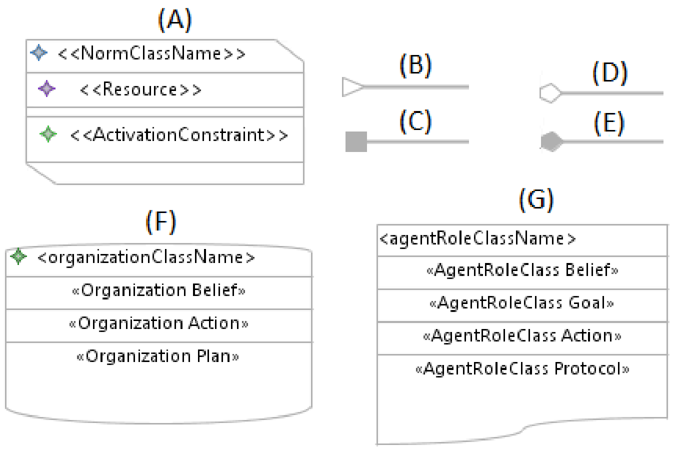
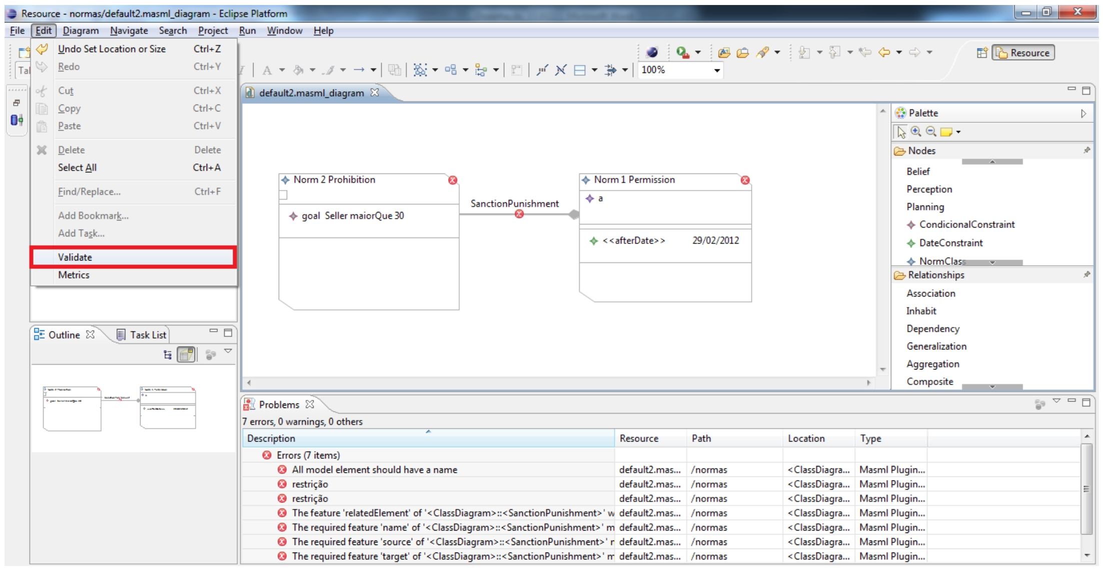


Figure 22: New representation of MAS-ML tool entities.



Problems

7 errors, 0 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (7 items)				
All model element should have a name	default2.mas...	/normas	<ClassDiagra...	Masml Plugin...
restrição	default2.mas...	/normas	<ClassDiagra...	Masml Plugin...
restrição	default2.mas...	/normas	<ClassDiagra...	Masml Plugin...
The feature 'relatedElement' of '<ClassDiagram>::<SanctionPunishment>' w	default2.mas...	/normas	<ClassDiagra...	Masml Plugin...
The required feature 'name' of '<ClassDiagram>::<SanctionPunishment>' m	default2.mas...	/normas	<ClassDiagra...	Masml Plugin...
The required feature 'source' of '<ClassDiagram>::<SanctionPunishment>' r	default2.mas...	/normas	<ClassDiagra...	Masml Plugin...
The required feature 'target' of '<ClassDiagram>::<SanctionPunishment>' m	default2.mas...	/normas	<ClassDiagra...	Masml Plugin...

Figure 23: Dashboard of Eclipse to generate the NorMAS-ML tool.

6. Example of Modeling

This section presents an example of modeling in order to illustrate our extension. We modeled the context of Conference Management System (Zambonelli *et al.*, 2001; Dignum, 2004; Harmon *et al.*, 2008) using NorMAS-ML. The role, organization and norm diagrams presented in this section were modeled using NorMAS-ML tool.

6.1. Conference Management System

Conference management systems are used to choose papers will publish in a Conference. Authors should submit theirs papers over a deadline. After this date, the reviewers should start the review process. In this process, each paper is reviewed for at least three reviewers that will analyses these papers considering the following aspects: originality, soundness, relevance, significance, quality of presentation, and understanding of the state of the art. After the finish of review period, the organizer will publish the results to authors and who has accepted paper must register in the conference.

Authors can: (i) submit their paper over deadline, (ii) view the paper status, and (iii) view the reviewers' comments after the finish of review process. On the other hand, reviewers can: (i) submit their paper over deadline, and (ii) evaluate the papers attributed to themselves by organizer. In their turn, organizers can: (i) extend the deadline, (ii) chose the reviewers to evaluate papers, and (iii) publish the review results.

Authors can submit two kinds of papers: (i) full papers that have a real contribution for research area, and (ii) short papers that are working with a preliminary result.

6.2. Entities of the System

In *Conference Management environment* is possible to identify the main organization called *Conference organization* and the *user agent* type that can play the following roles: *author*, *speaker*, *organizer*, *conference chair*, *website manager*, and *reviewer*. These agent roles were defined by main organization along with the object role called *submitted*. The instances of submitted can be played by instances of *Paper class* and its subclasses *ShortPaper* and *FullPaper class*. Figure 24 presents the class diagram specifying the relationship between classes and environment.

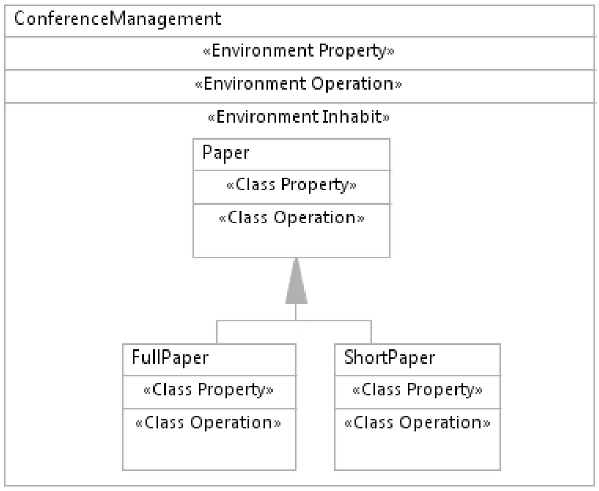


Figure 24: Class Diagram for Conference Management System.

Figure 25 shows the organization diagram of *Conference organization*. Because of this, classes of main organization along with classes of *agent*, *object*, *agent role* and *object role* were defined.

In order to allow the restriction of the behavior of entities related to an organization by norms, a set of restricted actions in organization must be defined explicitly. Figure 26 presents the detail of new representation of *Conference organization* updated after the elimination of << *axiom* >> stereotype.

Figure 27 presents the *role diagram* identifying the roles played by agents and objects in *Conference organization*. In this diagram *agent conference chair, website manager*, and *reviewer roles* are specializations of *organizer agent role*.

Figure 28 shows the detail of *Reviewer agent role* where norms are used to restrict the behavior of entities related to a determinate role, and a set of restricted actions in role entity is defined. These norms replace duties and right eliminated in NorMAS-ML definition.

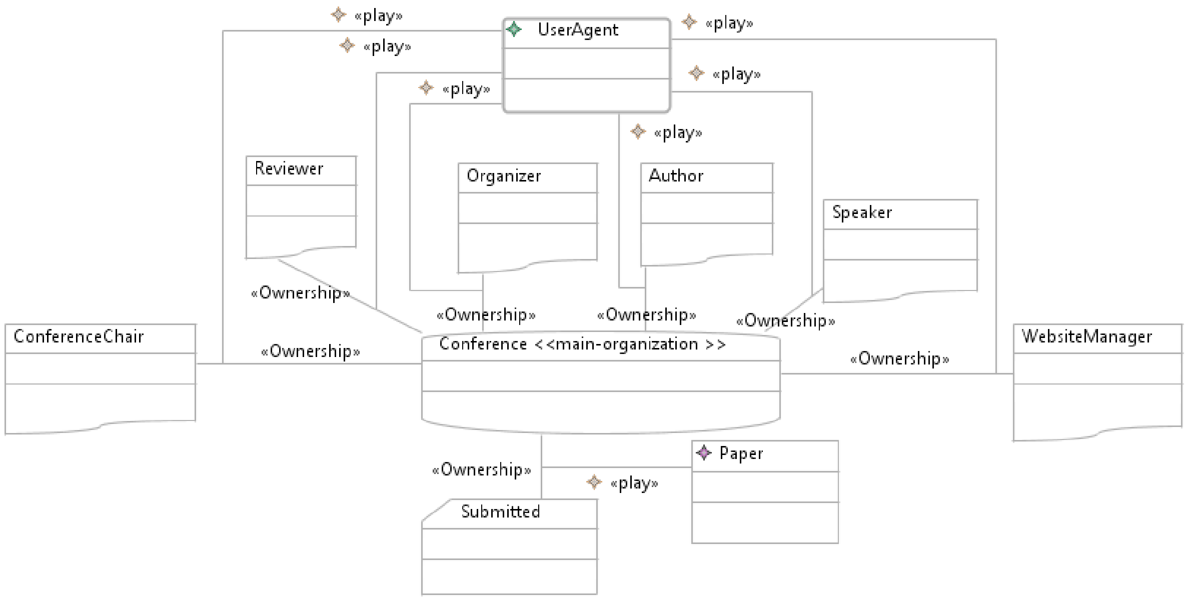


Figure 25: Organization Diagram for Conference Management System.

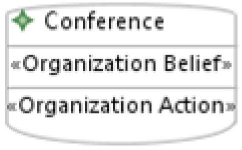


Figure 26: Conference Organization modeled with NorMAS-ML.

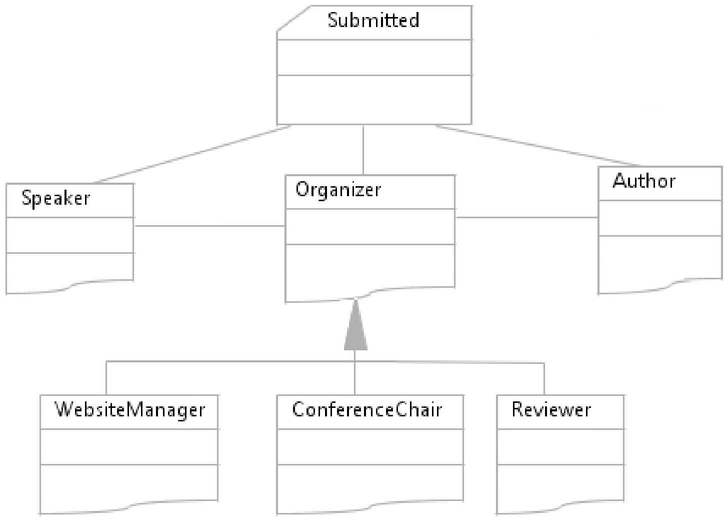


Figure 27: Role diagram for Conference Management System.

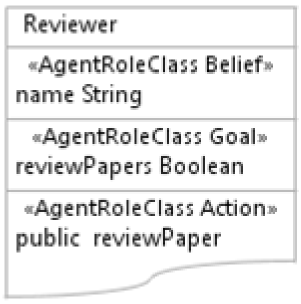


Figure 28: Reviewer agent role modeled with NorMAS-ML.

6.3. Norms of the System

Figueiredo (Figueiredo, 2011) defined a set of eleven norms for *Conference Management System*. From this set of norms: (i) six norms are obligations, two are prohibitions and three are permissions; (ii) one restricts a communicative action; (iii) five are activated by the achievement of systems states and two are activated by dates; and (iv) four are sanctions. Below, for each norm and its static components are presented together.

- N1 defines that *organizer agent roles* (involved entity) of *Conference organization* (context) are *prohibited* (deontic concept) *to submit paper* (non-communicative action);
- N2 describes that *reviewer agent roles* (involved entity) of *Conference organization* (context) are *permitted* (deontic concept) *to submit paper* (non-communicative action);
- N3 defines that *Conference Chair agent roles* (involved entity) of *Conference organization* (context) are *permitted* (deontic concept) to *extend the submission deadline* (non-communicative action) *if number of papers received is less that 50* (activation constraint);
- N4 describes that *reviewer agent roles* (involved entity) of *Conference organization* (context) are prohibited (deontic concept) *to review paper* (non-communicative action) *if author of name is equals to name of Reviewer* (activation constraint) and applies *N5 and N6 whether it is violated* (sanction - punishment);
- N5 defines that *reviewer agent roles* (involved entity) of *Conference organization* (context) are *obligated* (deontic concept) *to cancel Reviewer role* (non-communicative action) *when N4 is violated* (activation constraint);
- N6 describes that *Conference Chair agent roles* (involved entity) of *Conference organization* (context) are *obligated* (deontic concept) to discard paper (non-communicative action) *when N4 is violated and author of paper is equals to name of Reviewer that violates N4* (activation constraint);
- N7 defines that *reviewer agent roles* (involved entity) of *Conference organization* (context) are *obligated* (deontic concept) to review papers (non-communicative action) *before the notification deadline* (activation constraint);
- N8 describes that *Conference Chair agent roles* (involved entity) of *Conference organization* (context) are *obligated* (deontic concept) *to send an author notification* (communicative action) *if notification deadline* (activation constraint);
- N9 defines that *author agent roles* (involved entity) of *Conference organization* (context) are *obligated* (deontic concept) *to register at the conference* (non-communicative action) *before the registration deadline and if author have paper accepted* (activation constraint) and *applies N10* (sanction - punishment) or *N11* (sanction - reward) *whether it is violated or fulfilled*, respectively;

- N10 describes that *Conference Chair agent roles* (involved entity) of *Conference organization* (context) are *obligated* (deontic concept) to exclude the paper (non-communicative action) if *N9 is violated and if author of name is equals to name of author that violates N9* (activation constraint);
- N11 defines that *author agent roles* (involved entity) of *Conference organization* (context) are permitted (deontic concept) to *commit role speaker* (non-communicative action) *if N9 is fulfilled* (activation constraint).

The *Conference organization* is represented as an *Organization class* and the *ConferenceManagement environment* is represented as an *Environment*. The *Conference organization* belongs to the *ConferenceManagement environment* is represented by an *inhabit relationship*. *Conference* is composed of the *Organizer agent role* (agent roles are also represented as an *agent role class*) that is extended by the *Reviewer* and *ConferenceChair* agent roles (represented by the *generalization relationship* (See Figure 27)). As a result, *Reviewer* and *ConferenceChair* implicitly inherit the agent action *submitPaper* defined in *Organizer* (represented as an attribute of the type *agentAction*).

The *Reviewer agent role* has an agent action called *reviewPaper*, a belief called *name* and a goal called *reviewPapers*. Beliefs and goals are represented as attributes of the type *belief* or *goal* respectively. Norms are represented as a *Norm class* and a stereotype describing its deontic concept (e.g. << *Prohibition* >>, << *Permission* >>, or << *Obligation* >>).

All norms are defined in the context of the *Conference organization* (represented by a *Context relationship*). For each norm, we modeled its components using the *Norm diagram* defined in NorMAS-ML. N1 restricts the behavior of the *Organizer agent role* (represented by *restrict relationship*) stating a prohibition to the execution of the action *submitPaper* (represented as an attribute with the stereotype *«AtomicExecute»*, the entity (*Organizer*) and the action in intermediate compartment of *Norm*). N2 restricts the behavior of the *Reviewer agent role* stating a permission to the execution of the same action. Figure 29 shows the modeling of the norms N1 and N2 in Norm diagram.

Figure 30 shows the modeling of the norm N3, N7 and N8 in *Norm diagram*. The *Conference* is modeled as an *class* and has the attribute *numberOfPapers* of the type *Integer*. The papers of the conference are modeled as an class called *Paper* and it has the attribute author of the type String.

ConferenceChair agent role has an agent action called extendSubmissionDeadline and a protocol called *authorNotificationProtocol* which contains the message *authorNotification*. An *agentAction* is represented as an attribute in intermediate compartment in *Agent Role Class* while protocols are represented as an attribute in inferior compartment. N3 restricts the behavior of the *ConferenceChair agent role class* by stating a permission to the execution of the action *extendSubmissionDeadline* if the number of papers of the conference is < 50 (represented as an attribute in inferior compartment of N3, the stereotype *lessThan*, and the initial value 50) and if it is 21/02/2011 that is the date of the submission deadline (represented as an attribute with the stereotype << *equalTo* >>).

Figure 31 shows the *Norm diagram* containing the norms N4, N5, and N6. N4 restricts the behavior of the *Reviewer agent role* stating a prohibition to the execution of the reviewPaper agent action if the author of the paper is equal to its name (represented as an attribute with the << *equalTo* >> stereotype and the initial value referring to the name *belief* of the *Reviewer agent role*).

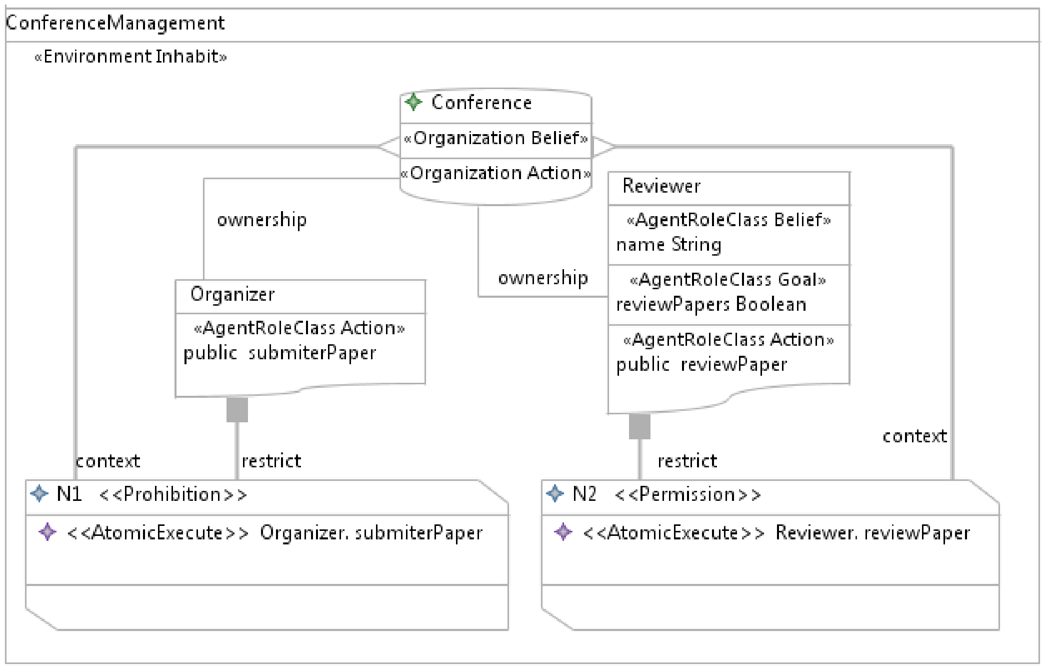


Figure 29: Modeling of the norms N1 and N2 with NorMAS-ML

N5 restricts the behavior of the *Reviewer agent role* by stating an obligation to cancel the *Reviewer agent role* (represented as an attribute with the stereotype << *AtomicCancel* >> and this agent role) as a sanction punishment for the violation of the norm N4. N6 restricts the behavior of the *ConferenceChair agent role* by stating an obligation to delete the paper (represented as an attribute with the stereotype << *atomicDelete* >> and this agent role) if the author of the paper is equal to the name of the reviewer (represented as an attribute with the stereotype << *equalTo* >> and the initial value referring to the name *belief* of the *Reviewer agent role*). N6 is defined as a punishment for the violation of the norm N4. Both N5 and N6 are sanctions (punishment) of N4 and for this reason their classes have *SanctionPunishment relationship* with N4.

N7 (see Figure 30) restricts the behavior of the *Reviewer agent role* by stating an obligation to the achievement of the goal *reviewPapers* (represented as an attribute with the stereotype << *atomicAchive* >> and this goal) before the date 31/03/2011 that is the date of the notification deadline (represented as an attribute with the stereotype << *beforeDate* >>).

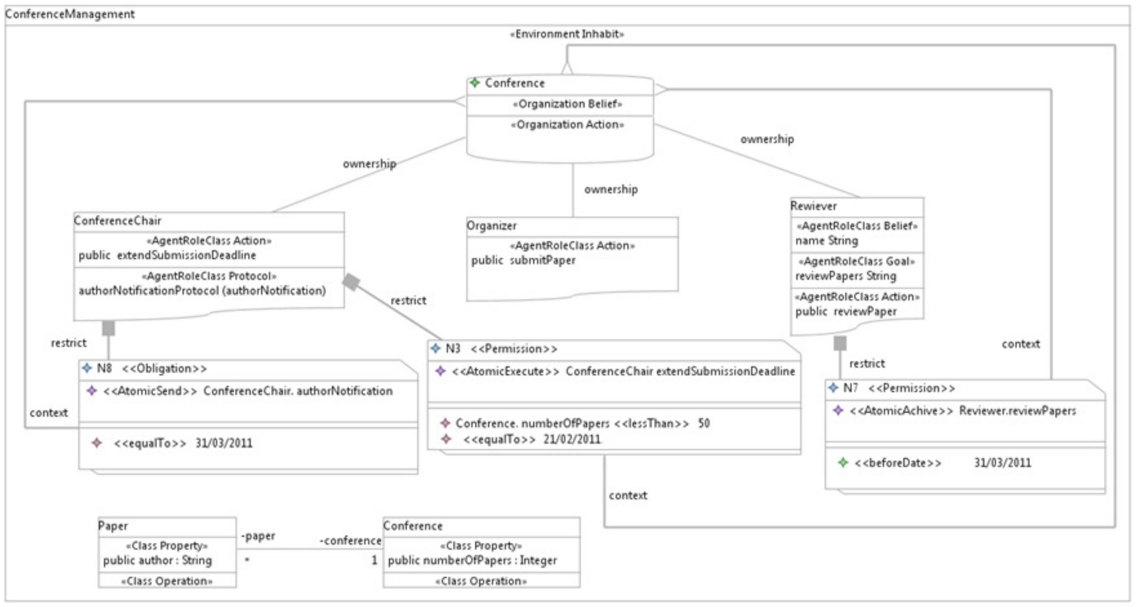


Figure 30: Modeling of the norms N3, N7, and N8 with NorMAS-ML.

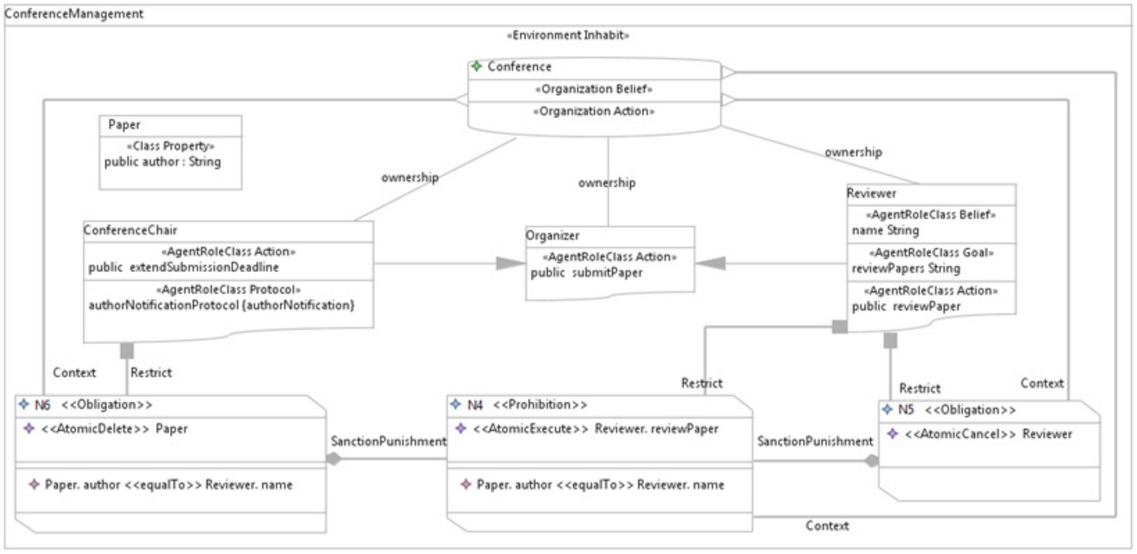


Figure 31: Modeling of the norms N4, N5, and N6 with NorMAS-ML.

In its turn, N8 (see Figure 30) restricts the behavior of the *ConferenceChair* agent role by stating an obligation to send the message *authorNotification* (represented as an attribute with the stereotype << *atomicSend* >>, the agent role and its attribute) if it is 31/03/2011 (represented as an attribute with the stereotype << *equalTo* >>).

Figure 32 shows the Norm diagram for norms N9, N10, and N11. *Conference organization* is also composed of the *Author* and *Speaker agent roles*. The *Author agent role* has an agent action called *registerAtConference*, a belief called name, and a goal called *havePaperAccepted*. The norm N9 restricts the behavior of the *Author agent role* by stating an obligation to the execution of the action *registerAtConference* before the date 31/04/2011 (date of the registration deadline) if the *Author* achieved its goal *havePaperAccepted* (represented as an attribute with the stereotype << *equalTo* >> and the initial value true).

In case of violation of N9, N10 states a punishment by restricting the behavior of the *ConferenceChair* agent role as an obligation to delete the paper (represented as an attribute with the stereotype << *AtomicDelete* >> and this class) if the author of the paper is the same name of the author that violated N9 (represented as an attribute with the stereotype << *equalTo* >> and name attribute of Author agent role). N10 is a sanction (punishment) of N9 and for this reason this class has a *SanctionPunishment relationship* with N9.

In case of fulfillment of N9, N11 states a reward by describing a permission to the *Author agent role* to commit with the *Speaker agent role* (represented as an attribute with the stereotype << *AtomicCommitment* >> and this agent role). N11 is sanction (reward) of N9 and for this reason this class has a *SanctionReward* relationship with N9.

Overall, we evidenced the capability of NorMAS-ML to model NMAS by that modeling example. Although we did adjustments in some entities of MAS-ML, NorMAS-ML is able to model all entities in MAS-ML, incorporating its set of MAS-ML diagrams and adding the *Norm diagram*.

7. Concluding Remarks

Analyzing organizational models and modeling languages for MAS, we noticed that they give a partial support to normative elements along with MAS entities. In this sense, we presented NorMAS-ML, a UML-based modeling language able to model the MAS main entities along with static normative elements. Besides, we defined NorMAS-ML tool supporting the NorMAS-ML concrete syntax.

Considering the static elements of norm, we presented TAO extension in order to define norms to regulate the behavior of TAO entities. Therefore, we defined *Norm* entity and its properties and a set of relationships that allow to relate this entity to other entities in TAO. After this, we defined the abstract and concrete syntaxes of NorMAS-ML and created a new static diagram called *Norm diagram*. In NorMAS-ML abstract syntax, we defined new metaclasses and stereotypes while graph elements were defined in NorMAS-ML concrete syntax.

In order to support the modeling of the *Norm diagram*, NorMAS-ML tool was defined by the evolving of MAS-ML tool. We defined OCL rules to check the consistency of models in the tool. We used NorMAS-ML tool to exemplify the concrete syntax of NorMAS-ML. Then, we modeled a Conference Management System considering all NorMAS-ML entities and relationships between them.

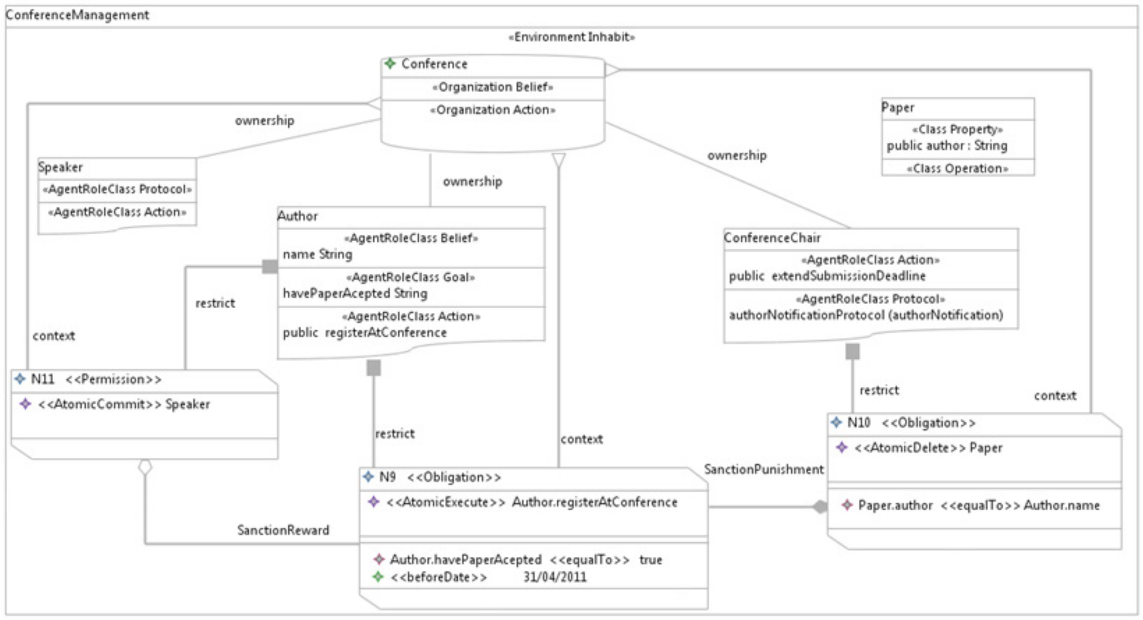


Figure 32: Modeling of the norms N9, N10, and N11 with NorMAS-ML.

This work has implications for researchers and practitioners. For researchers, NorMAS-ML is a good starting point for understand the relationship between MAS entities and norms, allowing the definition of examples to valid or discuss new approaches using MAS or NMAS. For practitioners, NorMAS-ML does not need other modeling languages for modeling NMAS, requiring the learning of one language syntax. Besides, the *Norm diagram* has a more complete view of MAS entities and static normative elements. This view can help software designers (i) to understand the properties and behavior of NMAS entities and (ii) to provide a software modeling following the stakeholders' need and less complex for the development phase.

As future work, we can suggest the inclusion of the verification of conflicts between norms and the code generation from NorMAS-ML models in NorMAS-ML tool and the extension of NorMAS-ML in order to support the modeling of the dynamic elements of norm, as creation, cancellation and delegation (da Silva Figueiredo *et al*., 2011). Besides, the use of design patterns can be explored while using NorMAS-ML for modeling other systems.

8. Acknowledgements

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 and by the Fundação Cearense de Apoio ao Desenvolvimento Científico e Tecnológico - Brasil (FUNCAP).

9. References

Basin, D., Doser, J., and Lodderstedt, T., 2006. Model Driven Security: From UML Models to Access Control Infrastructures. *ACM Trans. Softw. Eng. Methodol.*, 15(1):39–91. ISSN 1049-331X. doi:10.1145/1125808.1125810.

Bellifemine, F. L., Caire, G., and Greenwood, D., 2007. *Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology)*. John Wiley & Sons. ISBN 0470057475.

Boella, G., van der Torre, L., and Verhagen, H., 2006. Introduction to normative multiagent systems. *Computational & Mathematical Organization Theory*, 12(2):71–79. ISSN 1572-9346. doi:10.1007/s10588-006-9537-7.

Bordini, R. H., Hübner, J. F., and Wooldridge, M., 2007. *Programming Multi-Agent Systems in AgentSpeak Using Jason (Wiley Series in Agent Technology)*. John Wiley & Sons. ISBN 0470029005.

Braubach, L., Lamersdorf, W., and Pokahr, A., 2003. Jadex: Implementing a BDI-Infrastructure for JADE Agents.

Choren, R. and Lucena, C., 2005. *The ANote Modeling Language for Agent-Oriented Specification*, pages 198–212. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-31846-0. doi:10.1007/978-3-540-31846-0_12.

Czarnecki, K. and Eisenecker, U. W., 2000. *Generative Programming: Methods, Tools, and Applications*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA. ISBN 0-201-30977-7.

Danc, J., 2008. *Formal Specification of AML*. Master's thesis, Department of Computer Science Faculty of Mathematics, Physics and Informatics of Comenius University.

Dennis, L., Tinnemeier, N., and Meyer, J.-J., 2010. *Model Checking Normative Agent Organisations*, pages 64–82. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-16867-3. doi:10.1007/978-3-642-16867-3_4.

Dignum, V., 2004. *A model for organizational interaction: based on agents, founded in logic*. Ph.D. thesis, Universiteit Utrecht.

Eclipse, 2018. Eclipse Platform.

Farias, K., Nunes, I., Silva, V., and Lucena, C., 2009. MAS-ML Tool: Um Ambiente de Modelagem de Sistemas Multi-Agentes. In *Proceedings of the Fifth Workshop on Software Engineering for Agent-oriented Systems*.

Ferber, J., Gutknecht, O., and Michel, F., 2004. *From Agents to Organizations: An Organizational View of Multiagent Systems*, pages 214–230. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-24620-6. doi:10.1007/978-3-540-24620-6_15.

Ferber, J., Stratulat, T., and Tranier, J., 2009. Towards an Integral Approach of Organizations in Multi-Agent Systems. In Dignum, V., editor, *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, pages 51–75. IGI Global.

Figueiredo, K., 2011. *Modeling and Validation Norms in Multi-Agents Systems*. Master's thesis, Universidade Federal Fluminense, Instituto de Computação, Niterói, Brazil.

Figueiredo, K. and da Silva, V. T., 2011. Norm-ML - A Modeling Language to Model Norms. In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence - Volume 2*: ICAART, pages 232–237. ISBN 978-989-8425-41-6. doi:10.5220/0003179502320237.

Freire, E. S. S., Cortés, M. I., Gonçalves, E. J. T., and Lopes, Y. S., 2012. NorMAS-ML - A Modeling Language to Model Normative Multi-agent Systems. In Maciaszek, L. A., Cuzzocrea, A., and Cordeiro, J., editors, *ICEIS 2012 - Proceedings of the 14th International Conference on Enterprise Information Systems*, Volume 2, Wroclaw, Poland, 28 June - 1 July, 2012, pages 113–119. SciTePress.

Freire, E. S. S., Gonçalves, E. J. T., Cortés, M. I., Lopes, Y. S., and Brandão, M. G., 2013. TAO+: Extending the Conceptual Framework TAO to Support Internal Agent Architectures in Normative Multi-Agent Systems. *Electron. Notes Theor. Comput. Sci.*, 292:57-69. ISSN 1571-0661. doi:10.1016/j.entcs.2013.02.005.

García-Camino, A., Rodríguez-Aguilar, J.-A., Sierra, C., and Vasconcelos, W., 2006. Norm-oriented Programming of Electronic Institutions. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, pages 670–672. ACM, New York, NY, USA. ISBN 1-59593-303-4. doi:10.1145/1160633.1160750.

Gonçalves, E. J. T., Farias, K., Cortés, M. I., Feijó, A. R., Oliveira, F. R., and da Silva, V. T., 2011. MAS-ML Tool - A Modeling Environment for Multi-agent Systems. In Zhang, R., Cordeiro, J., Li, X., Zhang, Z., and Zhang, J., editors, *ICEIS 2011 - Proceedings of the 13th International Conference on Enterprise Information Systems*, Volume 2, Beijing, China, 8-11 June, 2011, pages 192–197. SciTePress.

Gonçalves, E. J. T., Cortés, M. I., Campos, G. A. L., Lopes, Y. S., Freire, E. S., da Silva, V. T., de Oliveira, K. S. F., and de Oliveira, M. A., 2015. MAS-ML 2.0: Supporting the modelling of multi-agent systems with different agent architectures. *Journal of Systems and Software*, 108:77 – 109. ISSN 0164-1212. doi:https://doi.org/10.1016/j.jss.2015.06.008.

Hannoun, M., 2002. *MOISE: un modèle organisationnel pour les systèmes multi-agents*. Ph.D. thesis, Ècole Nacionale Supèrieure des Mines de Saint-Etienne. Thèse (Doctorat).

Harmon, S. J., DeLoach, S. A., and Robby, 2008. *Trace-Based Specification of Law and Guidance Policies for Multi-Agent Systems*, pages 333–349. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-87654-0. doi:10.1007/978-3-540-87654-0_19.

Hübner, J. F., Sichman, J. S., and Boissier, O., 2002. *A Model for the Structural, Functional, and Deontic Specification of Organizations in Multiagent Systems*, pages 118–128. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-36127-5. doi:10.1007/3-540-36127-8_12.

Jennings, N. R., 1996. Foundations of Distributed Artificial Intelligence. chapter Coordination Techniques for Distributed Artificial Intelligence, pages 187–210. John Wiley & Sons, Inc., New York, NY, USA. ISBN 0-471-006750.

Jennings, N. R. andWooldridge, M., 2000. Agent-Oriented Software Engineering. ARTIFICIAL INTELLIGENCE, 117:277–296.

Lopes, Y., Cortés, M., Gonçalves, E. T., and Oliveira, R., 2018. JAMDER: JADE to MULTI-Agent Systems Development Resource. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 7(3). ISSN 2255-2863.

López y López, F., 2003. *Social Power and Norms: Impact on agent behavior*. Ph.D. thesis, University of Southampton, Faculty of Engineering and Applied Science, Department of Electronics and Computer Science.

Luck, M. and D'Inverno, M., 1995. *Structuring a Z specification to provide a formal framework for autonomous agent systems*, pages 46–62. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-44782-5. doi:10.1007/3-540-60271-2_112.

Meyer, J.-J. C. and Wieringa, R. J., editors, 1993. *Deontic Logic in Computer Science: Normative System Specification*. John Wiley and Sons Ltd., Chichester, UK. ISBN 0-471-93743-6.

Odell, J., Parunak, H., and Bauer, B., 2000. Extending UML for Agents. In *Proceedings of the Agent-Oriented Information Systems Workshop at the 17th National Conference on Artificial Intelligence*.

OMG, 2018. Unified modeling language.

Russell, S. J. and Norvig, P., 2003. *Artificial Intelligence: A Modern Approach. Pearson Education*, 2nd edition. ISBN 0137903952.

Silva, V., Garcia, A., Brandão, A., Chavez, C., Lucena, C., and Alencar, P., 2003. *Taming Agents and Objects in Software Engineering*, pages 1–26. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-35828-2. doi:10.1007/3-540-35828-5_1.

da Silva, V. T. and de Lucena, C. J. P., 2003. MAS-ML: A Multi-agent System Modeling Language. In *Companion of the 18th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, OOPSLA '03, pages 126–127. ACM, New York, NY, USA. ISBN 1-58113-751-6. doi:10.1145/949344.949383.

da Silva, V. T., Noya, R. C., and de Lucena, C. J. P., 2005. Using the UML 2.0 Activity Diagram to Model Agent Plans and Actions. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '05, pages 594–600. ACM, New York, NY, USA. ISBN 1-59593-093-0. doi:10.1145/1082473.1082563.

Silva, V. T. D., Choren, R., and Lucena, C. J. P. D., 2008. MAS-ML: a Multiagent System Modelling Language. *Int. J. Agent-Oriented Softw. Eng.*, 2(4):382–421. ISSN 1746-1375. doi:10.1504/IJAOSE.2008.020138.

da Silva Figueiredo, K., da Silva, V. T., and de Oliveira Braga, C., 2011. Modeling Norms in Multi-agent Systems with NormML. In *Proceedings of the 6th International Conference on Coordination, Organizations, Institutions, and Norms in Agent Systems*, COIN@AAMAS'10, pages 39–57. Springer-Verlag, Berlin, Heidelberg. ISBN 978-3-642-21267-3.

Sommerville, I., 2006. *Software Engineering*. Addison Wesley; 8th edition. ISBN 9780321313799.

Vasconcelos, W., Kollingbaum, M. J., and Norman, T. J., 2007. Resolving Conflict and Inconsistency in Norm regulated Virtual Organizations. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '07, pages 91:1–91:8. ACM, New York, NY, USA. ISBN 978-81-904262-7-5. doi:10.1145/1329125.1329236.

van der Vecht, B., Dignum, F., MEYER, J.-J., and DIGNUM, M., 2009. *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, chapter Autonomous Agents Adopting Organizational Rules. IGI Global. Pp. 314-333.

de Vries, W., Meyer, J. C., de Boer, F. S., and van der Hoek, W., 2009. A coordination language for agents interacting in distributed plan-execute cycles. *IJRIS*, 1(1/2):4–17. doi:10.1504/IJRIS.2009.026713.

Wagner, G., 2003. The Agent-object-relationship Metamodel: Towards a Unified View of State and Behavior. *Inf. Syst.*, 28(5):475–504. ISSN 0306-4379. doi:10.1016/S0306-4379(02)00027-3.

Wilber, K., 1997. An Integral Theory of Consciousness. *Journal of Consciousness Studies*, 4(1):71–92.

Zambonelli, F., Jennings, N. R., and Wooldridge, M., 2001. *Organisational Abstractions for the Analysis and Design of Multi-agent Systems*, pages 235–251. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-540-44564-7. doi:10.1007/3-540-44564-1_16.