8

An Approach to Operationalize Regulative Norms in Multiagent Systems

Carolina Howard Felicíssimo[1,2], Jean-Pierre Briot[2]
and Carlos José Pereira de Lucena[1]
[1]DI/PUC-Rio, Rua Marquês de São Vicente 225, 22453-900, RJ,
[2]LIPVI/ParisVI, Avenue du Président Kennedy, 75016, Paris
[1]Brazil
[2]France

1. Introduction

Multiagent systems have emerged as a promising approach to develop information systems that clearly require several goal-oriented problem-solving entities [Jennings *et al.*, 1998]. Following this direction, it is believed that upcoming information systems will be implemented as *open* multiagent systems, in which agents (their entities) can freely migrate among those systems in order to obtain resources or services not found locally.

A multiagent system(s) (hereinafter referenced as MAS[1]) is/are an example of an open system in which the actions of heterogeneous, self-interested agents may deviate from the expected behavior in a context. Openness has led to dynamic software systems that have no centralized control and that are composed of autonomous entities [Hewitt, 1991]. Key characteristics of such systems are heterogeneity, conflicting individual goals and limited trust [Artikis *et al.*, 2002].

As stated in [Esteva *et al.*, 2004], "*openness without control may lead to chaotic behavior*". In order to be a viable solution for dynamic software systems, MAS must be enhanced with norms for defining which actions are *permitted*, *obliged* and *prohibited* to be performed by agents so that the system does not reach an undesirable state. A *permitted* norm defines that an action is allowed to be performed; an *obliged* norm defines that an action must be performed; and, a *prohibited* norm defines that an action must not be performed.

Permissions and prohibitions are used to describe positive/negative authorizations, whereas obligations are used to describe responsibilities [Kagal and Finin, 2007]. These three types of norms represent the three fundamental *deontic statuses* of an action [Alberti *et al.*, 2006] from deontic logic [Wright, 1951] and they are logically connected as presented by the following statements:

- If an action is *permitted*, then, it is *not prohibited*;
- If an action is *obligatory*, then, it is *permitted* and it is *not prohibited*;
- If an action is *prohibited*, then, it is *not obligatory* and it is *not permitted*;

[1] Through all the text of this chapter, when the characteristic of open systems is important to be outlined, then, we will explicitly use the 'open MAS' expression instead of simply 'MAS'.

- If an action is *not permitted not to perform*, then, it is *obligatory*;
- If an action is *prohibited not to perform*, then, it is *obligatory*;
- If an action is *obligatory not to perform*, then, it is *prohibited*;

Deontic logic enables addressing the issue of explicitly and formally defines norms and deals with their possible violation [Alberti *et al.*, 2006]. In such way, deontic logic could be used in the agents' logics and architectures when norms can be violated and agents have to explicitly reason about those violations and their consequences [Jones and Sergot, 1993].

This means that agents should be able to take into account the existence of social norms in their decisions (either to follow or violate a norm) and to react to violations of the norms by other agents [Castelfranchi *et al.*, 1999].

Important works concerning normative MAS (*e.g.*, [Vázquez-Salceda *et al.*, 2005; Esteva, 2003; Hübner *et al.*, 2002; Minsky_LGI, URL; Chopinaud *et al.*, 2006]) have been proposed recently. A normative MAS is a system that conforms to or is based on norms [Boella *et al.*, 2006b]. That type of system must allow some facility for the system developer, while he is describing and evolving the norms of his system, and also allow some facility for agents' reasoning about applied norms.

We agree that current solutions for normative MAS usually have the following drawbacks: (i) they consider norms with a valid universal meaning in an application domain; (ii) they do not support the direct design and implementation of norms specific to the application domain (*e.g.*, political, economical, religious norms); (iii) they do not support the management of norms during system execution (*i.e.*, norm description off-line and norm enforcement on-line); and (iv) they expect that agents must be already aware of the (*predefined*) system norms.

The motivation for our research came forth from the need to resolve those challenges, providing an approach applicable in open systems, in which norms can be effectively applied to their agents and easily managed. In such systems, heterogeneity and autonomy rule out any assumption concerning the way third-party entities are implemented and behaved. Thus, a viable solution for regulation in MAS should not be hard coded inside agents' original implementations and must allow, for some degree of precision and flexibility, to update data (*e.g.*, norms) during the system execution.

Our research intends to bridge the gap between the theoretical work on norms and practical normative systems by proposing our DynaCROM approach [Felicíssimo *et al.*, 2008b and 2008c]. DynaCROM stands for *Dynamic Contextual Regulation Information Provision in Open MAS* and it aims to operationalize regulative norms in open MAS.

From the individual agents' perspective, DynaCROM is an information mechanism that makes application agents aware of the norms they are bound to at a given moment. From the system developers' perspective, DynaCROM is a methodology for the application and management of norms in open MAS so developers are able to embody abstract norms with domain values. Therefore, norms are contextualized in the application domain wherein they hold, facilitating regulation through norm enforcement mechanisms.

By 'context', DynaCROM follows the definition of [Dey, 2001] stating that "context is any implicit information that can be used to characterize the situation of participants and to provide relevant information and/or services to them, where relevancy depends on participants' tasks".

Regarding '*situation of participants*', DynaCROM is concerned with the issue of regulation in complex systems, which follows [Simon, 1996] in his definition stating that: "*complexity frequently takes the form of a hierarchy. That is, a system is composed of interrelated subsystems, each of which is in turn hierarchic in structure, until the lowest level of elementary subsystem is reached*".

In the remaining sections, the research involved to conceive our DynaCROM approach is presented. The theoretical fundamentals of DynaCROM are the main goal of the chapter. Further practical information related to the applicability of DynaCROM can be found via two motivating scenarios: in [Felicíssimo *et al.*, 2008a], through the domain of multinational organizations, and in [Felicíssimo, 2008d], through the television domain.

2. Dynamic contextual regulation information provision

Open MAS can be extremely dynamic due to heterogeneous agents that migrate among those systems for obtaining resources or services not found locally. In order to prevent malicious actions and to ensure agent trust, open MAS should be enhanced with normative mechanisms. However, it is not reasonable to expect that *foreign* agents will know in advance all the norms of the MAS in which they will execute.

In the following section, the DynaCROM methodology developed to support the system developer in the tasks of implementation, management and evolution of the norms of his MAS is explained. The methodology includes the phases of contextualization, concretization, representation and composition of norms.

2.1 From abstract to concrete norms in MAS

A major challenge in MAS is how norms can be effectively applied to their agents and, then, easily managed and evolved. The application of norms in MAS is not a straightforward task, since heterogeneity and autonomy rule out any assumption concerning the way that heterogeneous agents are implemented and behave in MAS [Grizard *et al.*, 2006].

In [Gaertner *et al.* 2007], the authors of the paper propose to extend the coordination level of a MAS with a normative level, so that, norms can be integrated during the design and execution time of the system. DynaCROM follows their proposition but, furthermore, it also proposes to extend the normative level with, what we called, a *contextual normative level*. In this level, abstract norms are concretized (*i.e.*, embodied) with domain values according to the context wherein they hold. The proposition for contextual classification of norms follows the ideas first proposed by Dignum in [Dignum, 2002] and, then, refined in [Grossi and Dignum, 2004]. However, their works mainly address formal issues while our approach addresses the practical ones, providing DynaCROM – an implemented solution as a proof-of-concept for the ideas proposed.

In order to illustrate the DynaCROM proposal, Figure 1 presents the *Coordination*, *Normative* and *Contextual Normative Levels* of a simplistic supply-chain scenario in which activities (illustrated by linked ellipses) are represented on the three layers (connected by dashed arrows). Norms (illustrated by vertical arrows) are applied at the second and last levels. Contextual norms (illustrated by diagonal arrows) are applied at the last level.

In order to exemplify how norms are concretized in normative levels, the *Negotiation* activity of Figure 1 is considered. A *Negotiation* summarizes a set of more specific activities performed between customer and seller agents (*e.g.*, a customer asks a seller how much a product costs; the seller states his price, with or without discounts; the customer accepts the seller's price). The *Negotiation* activity is linked to the *Payment* one and both activities might be translated in the normative level to:

A Payment Norm for Effecting a Negotiation: Negotiations are obliged
to be paid by using the national currency of the seller's country.

The payment norm presented above is abstract and vague, and therefore, applied only for general purposes. In order to cause any effect in a regulated system, abstract norms must be translated into concrete norms [Grossi and Dignum, 2004]. Thus, the abstract payment norm might be contextualized, by the system developer, as an environment norm and, then, concretized in its MAS. For example, in *American* and *Japanese* supply-chain domains, the environment norm could be concretized with the following instantiations:

A Concrete Environment Norm for Effecting a Negotiation: Negotiations are obliged to be paid (i) in *USA*, with *American dollars* (*USD*); and, (ii) in *Japan*, with *Japanese Yen* (*JPY*).

In the DynaCROM contextual normative level, the classificatory reading of '*counts-as*' from [Grossi *et al.*, 2006] is applied. The reading states that if "A counts-as B in context c", then, it is interpreted as "A is a sub-concept of B in context c". In this sense, '*counts-as*' statements work as '*contextual classifications*'.

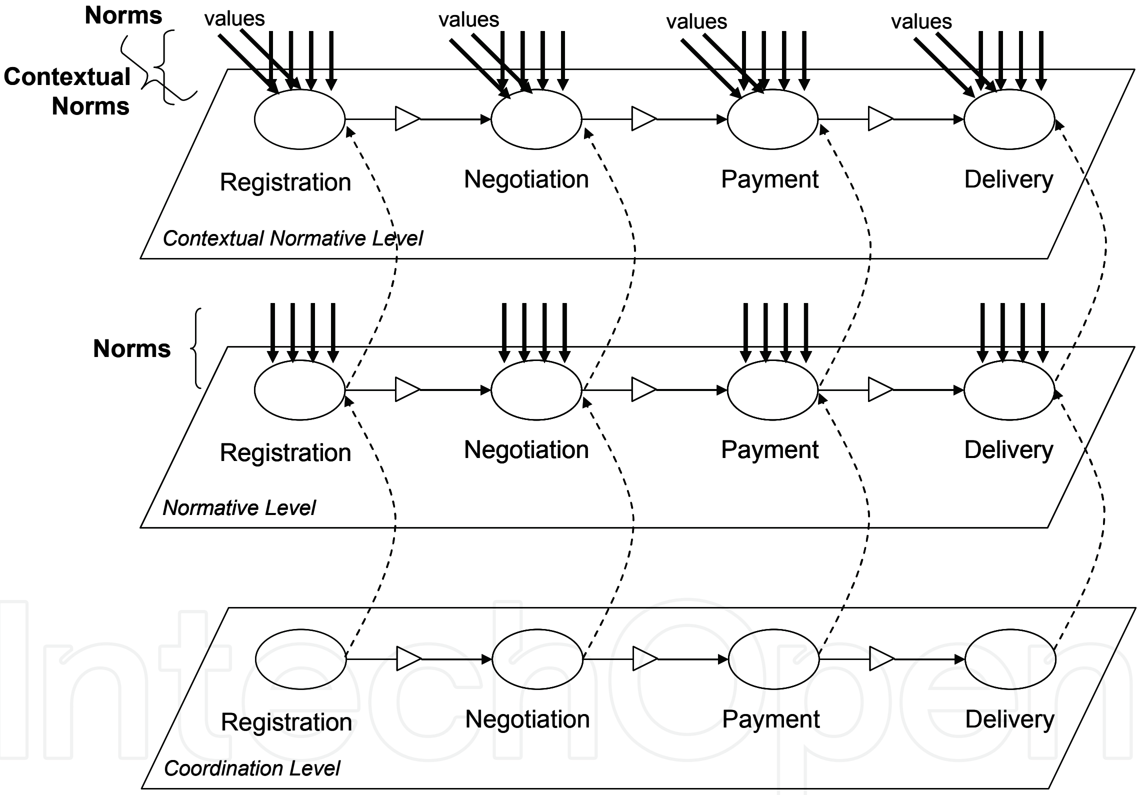


Fig. 1. Activities, regulated activities and contextual regulated activities represented in their specific layers, based on [Gaertner et al., 2007]

For instance, considering the payment norm exemplified above, its reading is done as follows: "*USD* counts-as a *valid currency* in the context of the *USA environment*"; and, its interpretation is done as follows: "*USD* is a sub-concept of a '*valid currency*' concept in the context of the *USA environment*".

Figure 2 illustrates part of the *Contextual Normative Level* of Figure 1 in which the '*valid currency*' variable of the *Negotiation* activity is instantiated for the payment norm according to the *American* and *Japanese* environments.

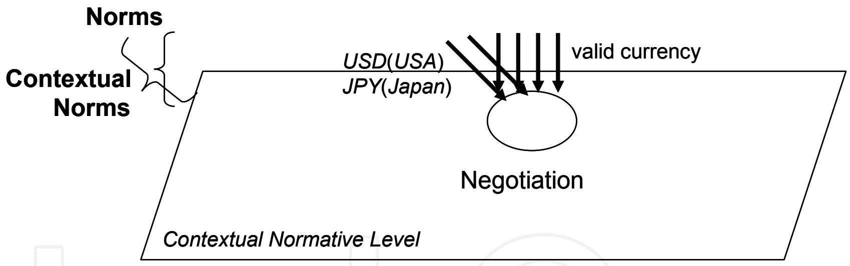


Fig. 2. Contextual classifications for a payment norm

Moreover, besides the instantiation of contextualized variables, DynaCROM consideres that activities, in the contextual normative level, can also have different predefined conditions. For instance, *Give discount* (a possible sub-activity of *Negotiation*) states that, in an organization, discounts can be given (a) by subtracting *10%* of the price value for *orders paid in cash*, or (b) by subtracting *15%* of the price value for *products bought in bundles*.

2.2 Contextual norm classification

Basically, a MAS consists of environments, organizations and agents playing roles and interacting [Jennings, 2000]. As environments, organizations, roles and agent interactions are important concepts for the understanding of the text, the meaning in which they are used in this chapter is characterized below.

Environments [Weyns *et al.*, 2007] are discrete computational locations, similar to places in the physical world, which provide conditions for agents to inhabit it. Environments can have refinement levels, such as a specialization relationship (*e.g.*, country and state), but there cannot be overlaps (*e.g.*, there cannot be two countries in the same place). An environment can also have many organizations.

Organizations [Ferber *et al.*, 2003] are social locations in which groups of agents play roles. An organization can embody many sub-organizations, but each organization belongs to only one environment [Silva and Lucena, 2004b]. Agents can execute in different organizations and they can also migrate among environments and organizations in order to obtain resources or services not found locally.

Roles [Thomas and William, 2005] are abstractions that prescribe a set of related tasks, which agents must perform in order to achieve their designed goals. Roles are defined by organizations independently of agents' individual identities.

An agent can interact with any other agent in a MAS, for example, by exchanging messages.

Environments, organizations, roles and interactions suggest different contexts for regulation in MAS. *Context-aware computing* means to be software that "*adapts according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time.*" [Schilit and Theimer, 1994]. Although contexts are tacitly known by most people, they are normally hard to be identified and, therefore, not distinguishable for computing.

In order to help the system developer in his task of norm contextualization, DynaCROM follows directions taken by research in context-aware applications that suggest top-down architectures for classifying contextual information [Khedr and Karmouch, 1995; Henricksen and Indulska, 2005].

DynaCROM defines that norm information should be classified in a MAS according to the following contexts: *Environment*, *Organization*, *Role* and *Interaction*, which are differentiated by the boundaries of their data (*i.e.*, norms).

Environment Norms are applied to all entities in a regulated environment. Likewise, *organization norms* are applied to all entities in a regulated organization; *role norms* are applied to all agents playing a regulated role; and, *interaction norms* are applied to all agents involved in a regulated interaction.

Figure 3 illustrates an example scenario in which entities of a MAS are influenced by the application of norms (represented by arrows) from different levels of abstractions (represented by dashed boxes). In the *Environment Normative Level* (upper level), environment norms are directly applied to environment instances (*e.g., USA* and *Japan*). In the *Organization Normative Level*, organization norms are directly applied to organization instances (*e.g., Dellie, HPie* and *DellieJapan*); in the *Role Normative Level*, role norms are directly applied to role instances (*e.g., ADellieSeller, AHPieSupplier* and *ADellieJapanManufacturer*); finally, in the *Agent Normative Level*, interaction norms are directly applied to the agents that are interacting.

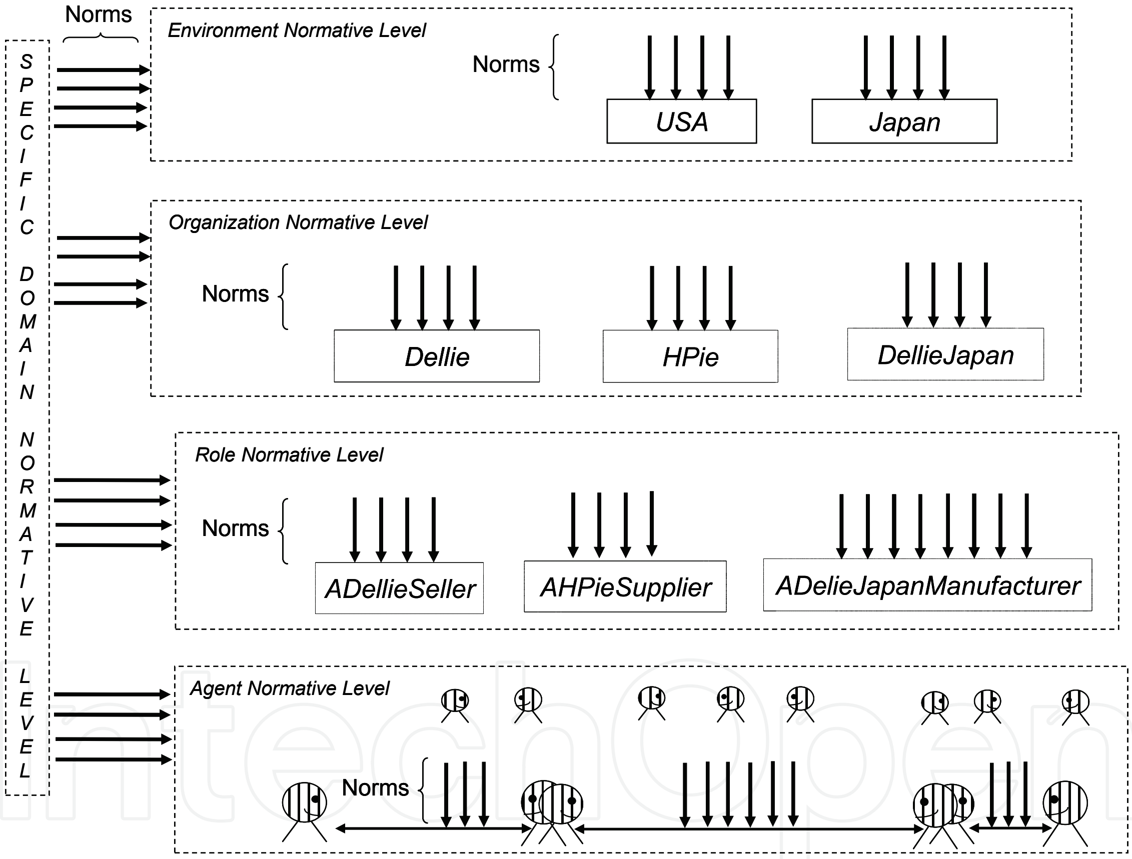


Fig. 3. Norms from different contextual normative levels

The four predefined normative contexts of DynaCROM are not targeted to a particular application domain; moreover, they rather represent a basic set for a general regulation in MAS. For a more precise regulation, this set should be improved through additions and refinements of application domain normative contexts and their respective norms. An example of a domain normative context and its norm might be, in the *Catholic* domain, a *Religious* concept that holds a (religious) norm stating that "*marriage is prohibited in the case that the man and/or the woman to be married made perpetual vows of chastity in a religious institute*".

DynaCROM states that specific[2] domain norms (*e.g.*, political norms) can be directly applied in any other normative level, as also illustrated in the Figure 5 (by the horizontal arrows from the vertical rectangle in the left side of the figure). For instance, the political norm: "*American and Japanese organizations are forbidden to deal with each other when their countries are undergoing political crisis*" can be directly applied in the *Organization Normative Level* of *American* and *Japanese* organizations (*e.g.*, in the *Organization Normative Level* of *Dellie* and *HPie*, from the *American* side, and in *DellieJapan*, from the *Japanese* side).

It is important to mention here that the extra effort of the system developer to classify norm information in more precise levels of abstraction is rewarded during the management phase and can also provide a fine-grained mechanism for norm enforcement solutions. Norms concretized in the contexts that directly affect the different MAS entities can be more easily found and updated because information is decoupled in predefined levels for norm classification. Besides that, each norm from the normative level of environments, organizations, roles or interactions can hierarchically influence each other, and each specific domain norm can transversally influence any norm from the other normative levels of a MAS.

2.3 Contextual norm representation

In order to represent norms in a meaningful way for heterogeneous agents, the formalism to be used in a MAS needs to be chosen. This choice must balance two major characteristics: *expressiveness* versus *efficiency*, and also should consider that, from a software engineering perspective, agent responses in MAS should be *quick*, *automatic* and *reliable* [Breitman *et al.*, 2004].

Speed is a requirement intrinsic to most systems. A *quick* response means that, once a request is sent by an agent, its response should be given at system runtime, even if the request may have been sent to multiple recipient agents, which compete for time of response.

Interoperability among agents in MAS lead to executions that must be *automatic*, *i.e.*, that cannot count with user intervention. One reason for that is, while the designer of a MAS is a domain expert, its human users may not be. Moreover, in open systems, a minimum level of *reliability* is mandatory in order to build trust for its participants.

The number of related norms involved in a negotiation among agents can be extremely high. In this case, it is not reasonable to expect that all system norms will be investigated in each negotiation, not in a reasonable time frame.

For the goal of our DynaCROM approach, it is accepted that the provision of a sub-set of relevant norms, where relevance is characterized by both agents' current contexts and actions performed, is a reasonable result if it is attained quickly, automatically and within reliable limits (predefined by the system developer). This way, information does not need to be stored since the relevant norms are provided, each time, at system/agents' requests.

Norm Representation by Using OWL

The *Web Ontology Language* (OWL) [Bechhofer *et al.*, URL], a Web standard from the *World Web Consortium* (W3C), was analyzed in order to verify its applicability for norm representation in open MAS. OWL was chosen mainly because of the two following reasons.

[2] 'Specific' meaning 'particular' and having 'general' as its antonym (definition from the Roget's New Millennium™ Thesaurus [OnLineDictionary, URL]).

The first reason is because OWL represents information in a meaningful way (*i.e.*, with a common understanding) for heterogeneous agents, supporting them in their processes of data retrieving and integration with different sources. This way, information can be understood by computer applications, instead of only by humans.

The second reason for choosing OWL is because it provides three sublanguages, which are differentiated by their levels of expressiveness: OWL Lite, OWL DL (includes OWL Lite) and OWL Full (includes OWL DL).

OWL Lite was designed for easy implementation and to provide users with a functional subset that permits a classification hierarchy and simple constraints.

OWL DL (where DL stands for *Description Logic*) was designed to support those users who want the maximum expressiveness without losing computational completeness (*i.e.*, all entailments are guaranteed to be computed) and decidability (*i.e.*, all computations will finish in finite time) of reasoning systems. OWL DL is so named due to its correspondence with description logics [Baader *et al.*, 2003], a field of research that has studied a particular decidable fragment of first order logic. OWL DL was designed to support the existing Description Logic business segment and to provide a language subset that has desirable computational properties for reasoning systems.

OWL Full is meant for users who want expressiveness with no computational guarantees. In this case, OWL Full relaxes some of the constraints on OWL DL so as to make available features which may be of use to many database and knowledge representation systems, but which violate the constraints of Description Logic reasoners. Thus, it is unlikely that any reasoning software will be able to support every feature of OWL Full.

Based on the characteristics of each OWL sublanguage, OWL DL was the one chosen for representing the domain data of the usage scenarios presented in this chapter. This is because, in those examples, OWL DL meets the software engineering requirements for responses in MAS (*i.e.*, expressiveness in computational completeness and decidability). Therefore, *formal* versus *non-formal* issues related to the examples are restricted to the available properties of the OWL DL sublanguage.

Declarative Specifications of Concrete Norms

DynaCROM proposes a *contextual normative ontology* for declarative specifications of norms, providing information with a common understanding about well-defined system regulation to heterogeneous agents.

An *ontology* is a conceptual model that embodies shared conceptualizations of a given domain [Gruber, 1993]; a *contextual ontology* is an ontology that represents localized domain information [Bouquet *et al.*, 2003] (*e.g.*, *USD* is the national currency of *USA*); and, a *contextual normative ontology* is a contextual ontology that has a *Norm* concept as its central asset. The *Norm* concept should be instantiated with norms contextualized differently according to the basic MAS entities (*i.e.*, environments, organizations, roles and agent interactions) or specific domain entities.

The DynaCROM contextual normative ontology, hereinafter the DynaCROM ontology, defines the following five related concepts, all in the same hierarchical level: *Role*, *Organization*, *Environment*, *Norm* and *Action*, as illustrated in Figure 4[3]. These concepts must be instantiated according to the application domain of its MAS.

[3] For readability purposes, all ontologies created for this chapter are presented graphically by using the Ontoviz graph plug-in for OWL [Ontoviz, URL].

In the DynaCROM ontology, the *Role* concept encompasses the instances of all regulated roles of the system and each role instance is associated with its norms (via the *hasNorm* property) and with its organization (via the *isPlayedIn* property). The *Organization* concept encompasses the instances of all regulated organizations and each organization instance is associated with its norms, with itself (via the *hasMainOrganization* property for representing its main organization) and with its environment (via the *isIn* property). The *Environment* concept encompasses the instances of all regulated environments and each environment instance is associated with its norms and with itself (via the *belongsTo* property for representing its owner environment). The *Norm* concept encompasses the instances of all norms and each norm instance is associated with its regulated actions (via the *regulates* property). The *Action* concept encompasses the instances of all regulated actions of a proposed system.

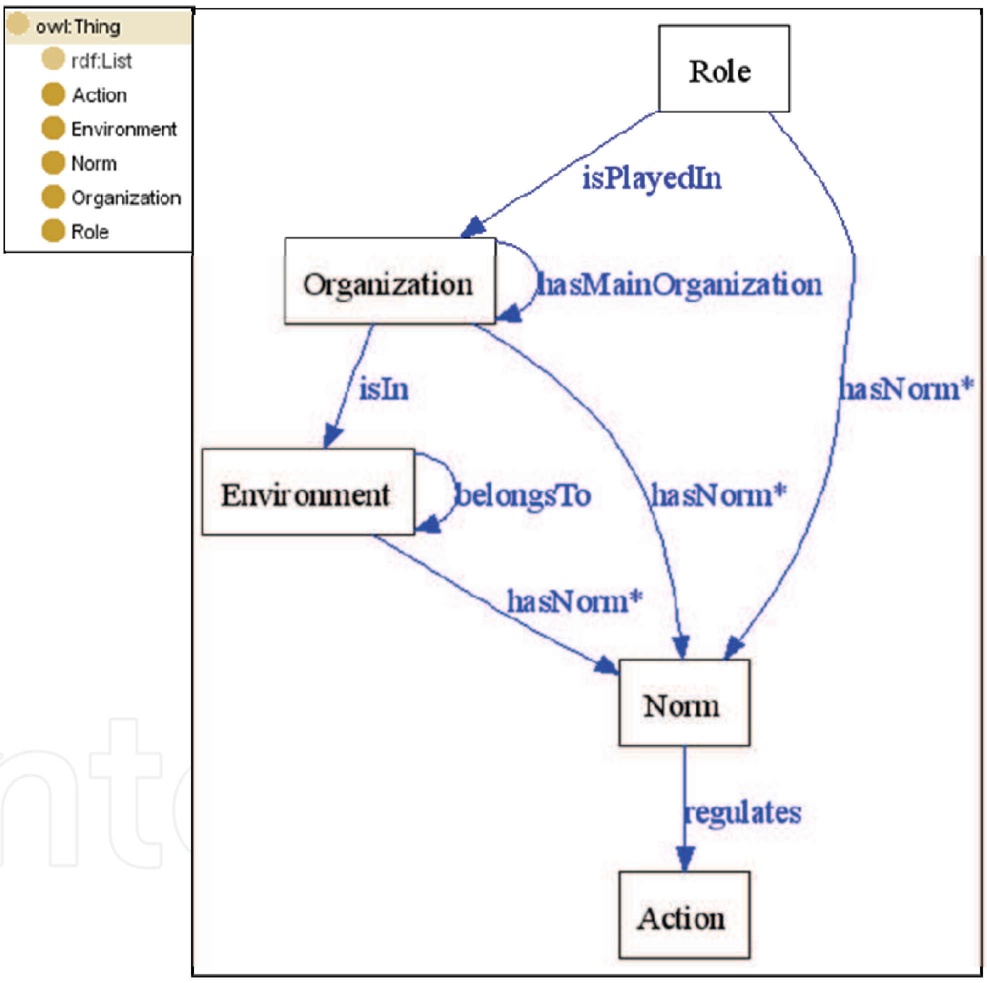


Fig. 4. The DynaCROM ontology

The DynaCROM ontology is an extensible one, *i.e.*, its basic concepts can be extended and/or new domain concepts can be created, both for representing classified contextual domain information. More precisely, the representation of a concrete norm in a DynaCROM ontology should be done by extending existing concepts or by creating new ones, then, instantiating the concept with norm information and, at last, linking the regulated instances to its related abstract norm (represented as a created norm instance).

For example, Figure 5[4] illustrates the *OblToPayWithNationalCurrency* norm instance that represents an abstract payment norm for effecting a negotiation. The norm is concretized in each environment by instantiating its domain datatype property *hasNationalCurrency* (*e.g.*, *JPY (Japanese Yen)* in *Japan* and *USD (U.S. Dollar)* in *USA*), which extends the DynaCROM *Environment* concept (originally presented in Figure 4).

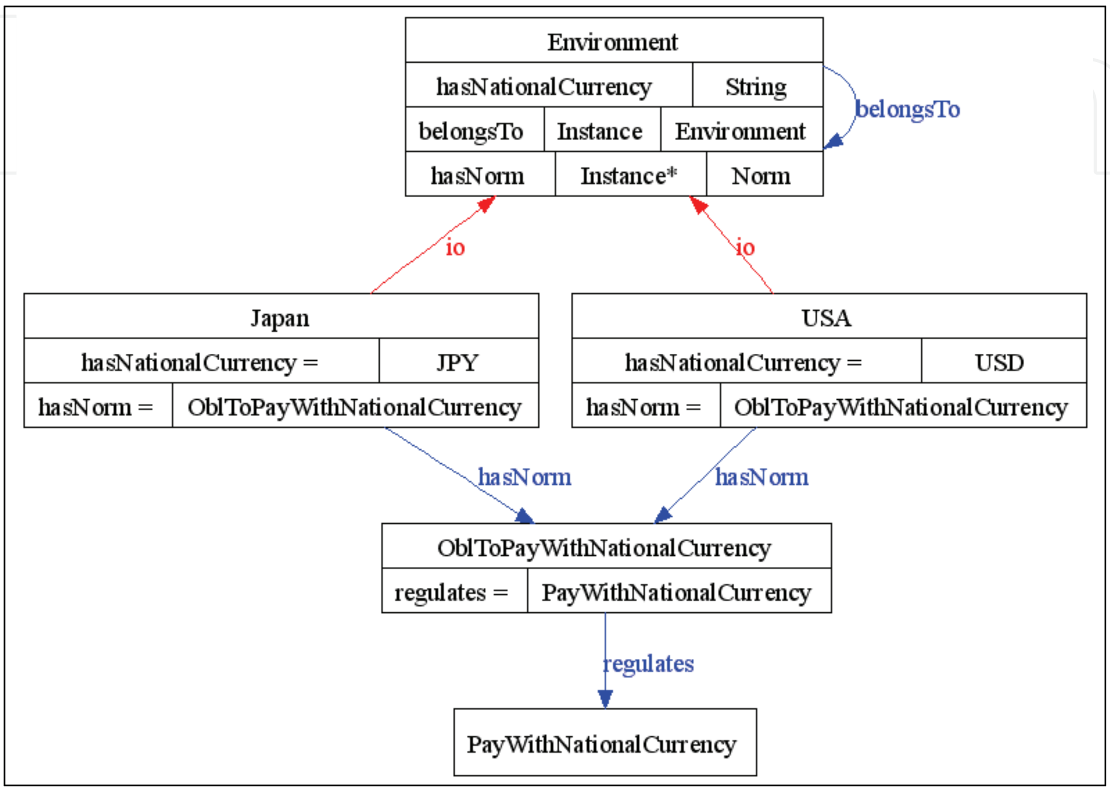


Fig. 5. An abstract payment norm concretized in *Japan* and *USA*

For a more precise regulation in MAS, specific domain contexts should be represented in an application domain DynaCROM ontology through additions and refinements of their related concepts and norms. An example of an application domain context and its norm can be:

> *A Political Norm for Regulating Deals:* organizations are prohibited from dealing with each other when their countries are undergoing political crisis.

The political norm presented above is an example of abstract interaction norm. In a DynaCROM domain ontology, interaction norms should be concretized by instantiating its *Norm* sub-concept, which must be already created for linking the other concepts from the relation (*i.e.*, reification of relationship). This solution follows the representation pattern presented in [Noy and Rector, URL].

Figure 6 illustrates the abstract political norm for regulating deals represented by the *PrhToDealWith* (a *Norm* sub-concept) and concretized in *AmericanOrganizations* by the *PrhToDealWithJapaneseOrganizations* norm instance. The concrete norm prohibits *AmericanOrgani-*

[4] In the Ontoviz graph plug-in, ‘io’ means ‘instance of’ and it is the label given for the link between a concept and its instance.

zations to deal with *JapaneseOrganizations* when their countries are undergoing political crisis.

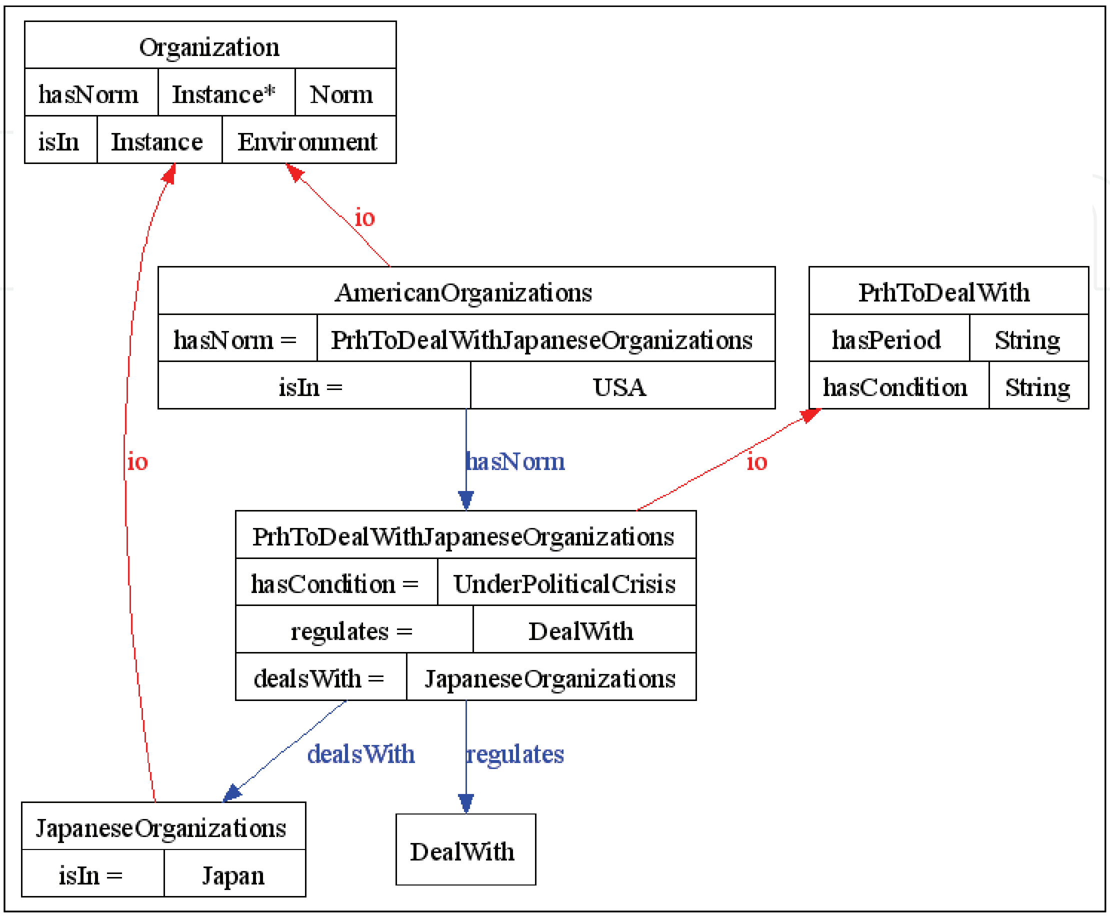


Fig. 6. A political norm concretized in *American* organizations

2.4 Contextual norm composition

After classifying and representing norms in precise levels of abstractions, contextual norms can be composed during system execution since, at any given moment, an agent may be related to norms defined at one or more normative contexts. Compositions of related contextual norms result in sets of independent norms, in which the semantic of one norm can influence the semantics of the others. For instance, the environment norm presented below is considered:

> *A Concrete Environment Norm for Calculating Prices:* a state corporate income tax rate of *6.25* in *Missouri* is obliged to be imposed on all sales.

Figure 7 illustrates the environment norm for calculating prices in *Missouri*. Although *Missouri* and *USA* are hierarchical environments (defined via their *belongsTo* relationship), a mechanism should be used in order to effectively compose their norms in a DynaCROM MAS.

DynaCROM follows rules to compose contextual norms. DynaCROM rules are *ontology-driven* rules, *i.e.*, they are created by the system developer, according to the ontology structure, and they are limited to the related concepts to which each concept is linked to.

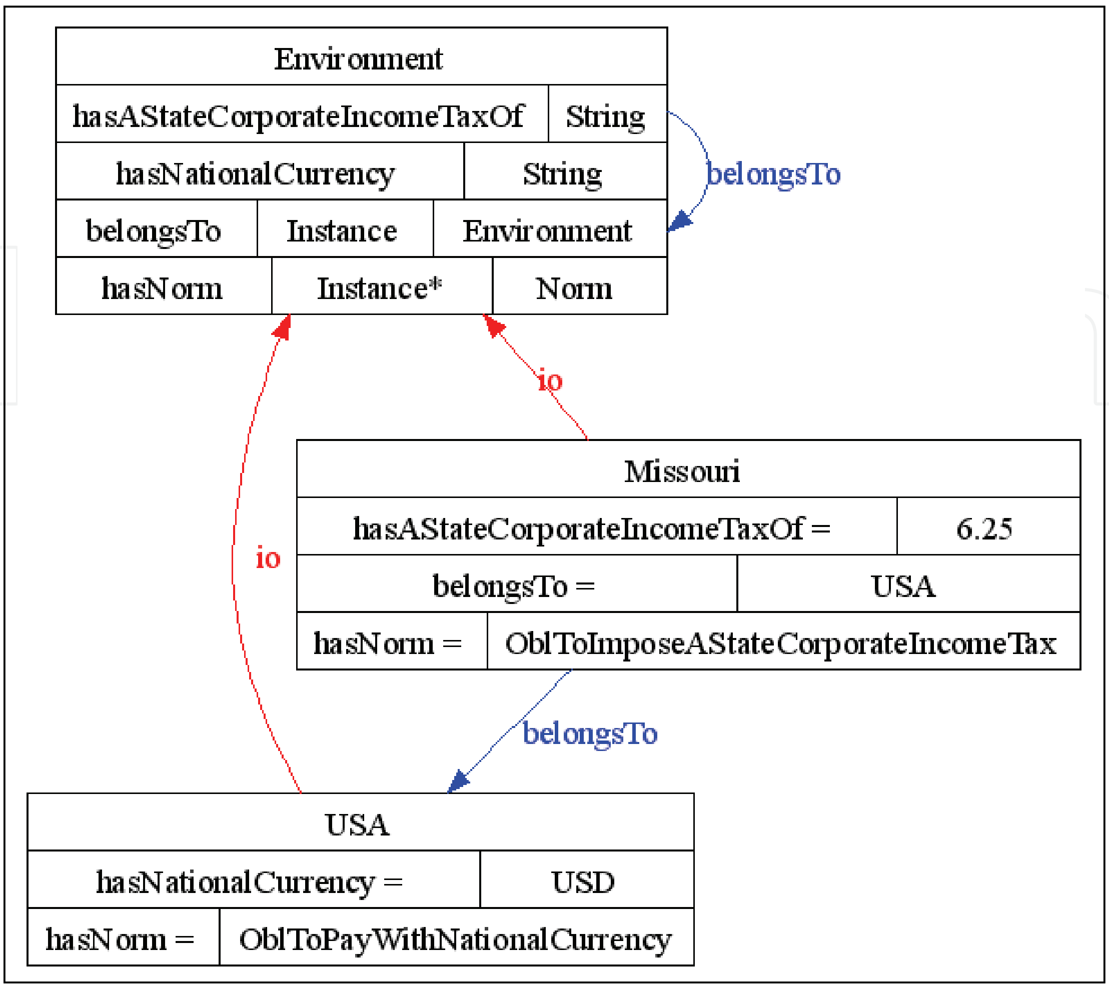

Environment
hasAStateCorporateIncomeTaxOf | String
hasNationalCurrency | String
belongsTo | Instance | Environment
hasNorm | Instance* | Norm
belongsTo
io
io
Missouri
hasAStateCorporateIncomeTaxOf = | 6.25
belongsTo = | USA
hasNorm = | OblToImposeAStateCorporateIncomeTax
belongsTo
USA
hasNationalCurrency = | USD
hasNorm = | OblToPayWithNationalCurrency

Fig. 7. The *Missouri* and *USA* hierarchical environments

Code 1[5] presents an example of rule that recursively compose the norms of hierarchical environments as, for instance, the norms of *Missouri* and *USA*. More precisely, considering *Missouri* as an example of the given environment, the following composition process is executed, according to the domain ontology instance illustrated in Figure 9: in (4), the '*?OEnv*' variable is instantiated with the *USA* inferred value, when the '*?Env*' variable is instantiated with the *Missouri* given value; in (3), the '*?OEnvNorms*' variable is instantiated with the *OblToPayWithNationalCurrency* inferred value; and in (2), the inferred norm is added as a new norm of *Missouri*.

The result of the norm composition process is that, in *Missouri*, all negotiations are obliged to be paid with *USD* and increased by a state corporate income tax of *6.25*.

```
(1)   [DynaCROMRule_EnvWithOEnvNorms:
(2)     hasNorm(?Env,?OEnvNorms)
(3)      <- hasNorm(?OEnv,?OEnvNorms),
(4)         belongsTo(?Env,?OEnv)]
```

Code 1. A DynaCROM rule to compose the norms of hierarchical environments

[5] The rules presented in this chapter are written following a simplified syntax for readability purposes.

DynaCROM predefines the rules to compose the norms of hierarchical environments (explained above) and also the others presented in Code 2. Inputs for these rules are domain instances of the *Organization* and *Role* concepts and their outputs are compositions of related contextual norms.

Following a norm composition process similar to the one explained for the '*DynaCROMRule_EnvWithOEnvNorms*' (presented in Code 1), the '*DynaCROMRule_OrgWithMOrgNorms*' (line 5 to 8 from Code 2) states that a given organization will have its norms composed with the norms of its main organization; the '*DynaCROMRule_OrgWithEnvNorms*' (line 9 to 12 from Code 2) states that a given organization will have its norms composed with the norms of its environment; and, the '*DynaCROMRule_RoleWithOrgNorms*' (line 13 to 16 from Code 2) states that a given role will have its norms composed with the norms of its organization.

```
(5)  [DynaCROMRule_OrgWithMOrgNorms:
(6)    hasNorm(?Org,?MOrgNorms)
(7)     <- hasNorm(?MOrg,?MOrgNorms),
(8)        hasMainOrganization(?Org,?MOrg)]

(9)  [DynaCROMRule_OrgWithEnvNorms:
(10)   hasNorm(?Org,?OrgEnvNorms)
(11)    <- hasNorm(?OrgEnv,?OrgEnvNorms),
(12)       isIn(?Org,?OrgEnv)]

(13) [DynaCROMRule_RoleWithOrgNorms:
(14)   hasNorm(?Role,?OrgNorms)
(15)    <- hasNorm(?Org,?OrgNorms),
(16)       isPlayedIn(?Role,?Org)]
```

Code 2. DynaCROM rules to compose the norms of normative contexts

Rules can compose data from the same concept type (*e.g.*, the '*DynaCROMRule_EnvWithOEnvNorms*' and the '*DynaCROMRule_OrgWithMOrgNorms*') or from different concept types (*e.g.*, the '*DynaCROMRule_OrgWithEnvNorms*' and the '*DynaCROMRule_RoleWithOrgNorms*'). Rules can also compose data from concepts directly related (hierarchical form) or indirectly related (non-hierarchical form).

Code 3 presents an example of a rule that compose the norms of the DynaCROM *Role* and *Environment* concepts, which are examples of indirectly related concepts.

```
(17) [DynaCROMRule_RoleWithOrgEnvNorms:
(18)   hasNorm(?Role,?OrgEnvNorms)
(19)    <- hasNorm(?OrgEnv,?OrgEnvNorms),
(20)       isIn(?Org,?OrgEnv),
(21)       isPlayedIn(?Role,?Org)]
```

Code 3. A rule for composing the norms of two indirectly related concepts

For the composition process, DynaCROM employs[6] an inference rule engine that executes the following tasks: (i) read an ontology instance to get data (*i.e.*, concept instances and their

[6] 'Employ' meaning "to make use of (an instrument, means, etc.); use; apply" [OnLineDictionary, URL]).

relationships), (ii) read a rule file to retrieve the information about how concepts must be composed; and then, (iii) infer an ontology instance based on the previous readings. An overview of this composition process is illustrated in Fig. 8.

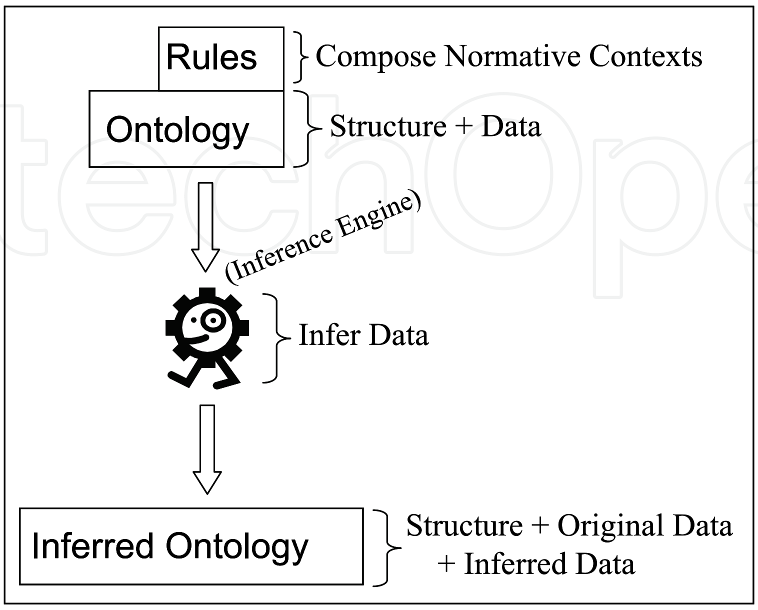


Fig. 8. The DynaCROM composition process

Once the domain ontology and/or rule file change(s), updated information is automatically forwarded to system agents in the next DynaCROM execution. This makes it possible for information management to be done at runtime, providing the dynamicity and flexibility necessary for regulation and also regarding social changes characteristic of MAS. These achievements are gotten because all norms provided by DynaCROM are applicable at a given moment.

In the current implementation of DynaCROM, the Jena rule-based inference engine [JENA, URL] is used for the composition process, however, other Semantic Web reasoners, like Racer [RACER, URL], Pellet [PELLET, URL] or FaCT [FACT, URL], can also be used.

In the composition process, it will still be the system developer's responsibility to write rules in the exact order he wants to compose his system data. Normally, the chosen inference engine will read/interpret those rules in sequence, from the top to the bottom of the file.

3. Contextual norm enforcement

DynaCROM is an approach for implementing dynamic MAS, in which norms can be updated at system runtime. In that way, agents are continuously supported with precise information about the current norms they are bound to in a given moment. Nevertheless, a regulated MAS should verify if a performed action is legal or illegal based on its defined norms, which might be enforced. However, it is the responsibility of the system developer to define if the norms of his MAS are allowed or not to be violated, by imposing the correct strategy for norm enforcement.

Norm enforcement in MAS can be carried out *a posteriori*, by punishing infringing agents, or *a priori*, by avoiding norm violation. *A posteriori* enforcement does not guarantee norm compliance, however, the implementation of punishments inhibits infringing agents. *A* *priori* enforcement guarantees norm compliance while enforcing the norms of the regulated actions of a MAS.

In the following two subsections, it is illustrated how DynaCROM supports the norm enforcement *a posteriori* and *a priori*, respectively. In subsection 3.3, an overview of the process on how DynaCROM works as an input mechanism to third-party enforcers is given. Finally, in the last subsection, it is exemplified how norms are enforced based on the agents' external and internal behaviors.

3.1 *A Posteriori* norm enforcement

The aim of any society and its norms is to provide a common space for the realization of individual and global objectives of its participants. Sometimes, depending on both the goals of agents and their priorities, the violation of norms is the best choice for agents (and also for the society). In this sense, norms act as a goal-oriented decision mechanism in regulated systems, being the means to achieve goals in a society. Hence, norms are allowed to be violated instead of being defined as constraints which unable any undesired behavior to happen.

In [Felicíssimo *et al.*, 2005b], it is given an example of a MAS, from the urban traffic domain, in which its norms might be violated. In the motivating scenario of the example, an agent playing a car driver role is going from his home in the city to his summer house on the mountains (see Figure 11). Suddenly, on the way, his pregnant wife begins to go into labor. Now, the goal of the driver agent to get to a hospital, as soon as possible, emerges as the one with the highest priority. Then, he considers violating some norms. In the emergency situation, the agent prefers to pay the fine for going through a red traffic light, if that will get him to the hospital faster. However, if pedestrians were crossing the street in front of the traffic light, then, the driver would not go through the red light. This is because of the risk of killing pedestrians and, thus, the fine he would have to pay would be too high. Both fines could be informed by DynaCROM, through requests from the car driver agent.

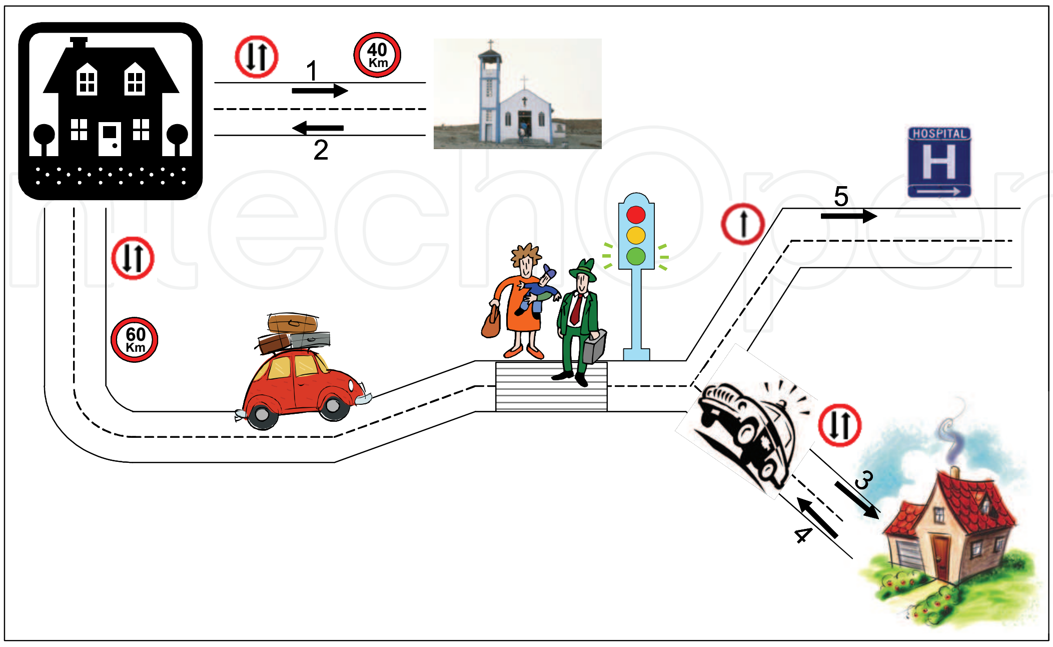


Fig. 9. An urban traffic's scenario

Another emergency situation which requires agent reasoning for norm violation is illustrated in Figure 12. An agent is driving down a road when he perceives that another driver agent is trying to pass his car on the left. Suddenly, a cow appears in front of his car. Now, what should the driver agent do? His decision must be based on the norms applicable to him, *i.e.*, on his contextual norms informed by DynaCROM.

If the situation of the motivating scenario occurred in an area in which the *Hinduism* religion is practiced, as in *India*, then, it would be better for the car driver to crash into the other car. This is because the fine of hurting a cow in a *Hindu* territory might be too high (*Hindus* believe that cows are sacred animals and, therefore, must be kept in safety). However, in *non-Hindu* countries, perhaps, it would be better to run over the cow instead of crashing into the other car. In both cases, if no risk to human lives was involved, then, the decision of the car driver would be based on how high a fine he would have to pay.

Fig. 10. A situation in which contextual norms must be considered

Sanction for *a posteriori* norm enforcement

A norm violation is a situation in which an agent breaks one or more norms, entering in an illegal (unsafe) state. In order to make agents legal again (*i.e.*, performing back in a safe state), sanction can be used in *a posteriori* norm enforcement. Sanction is a set of actions whose realization will remove the violation, by paying its consequences. In that way, agent are informed about the drawback of violating a norm. Sanction can be represented in the DynaCROM ontology inside a *Sanction* new concept, which holds the consequences of a violated norm.

Figure 13 illustrates an example that killing cows is prohibited in the *Hinduism* religion. In the example, the DynaCROM ontology was extended with the *Religion* and *Sanction* domain concepts; moreover, the DynaCROM *Environment* concept was extended with the domain object property *hasReligion*, for concretizing religions in each environment, and the DynaCROM *Norm* concept was extended with the domain object property *hasSanction*, for concretizing sanctions in each norm.

Figure 11 also illustrates the DynaCROM ontology instantiated for the example. The *India* instance represents the environment that holds the *Hinduism* religion, which, in turn, holds the *PrhToKillCows* prohibition norm. This norm regulates the *KillCows* action and has *A10YearPrisonSentence* as its sanction.

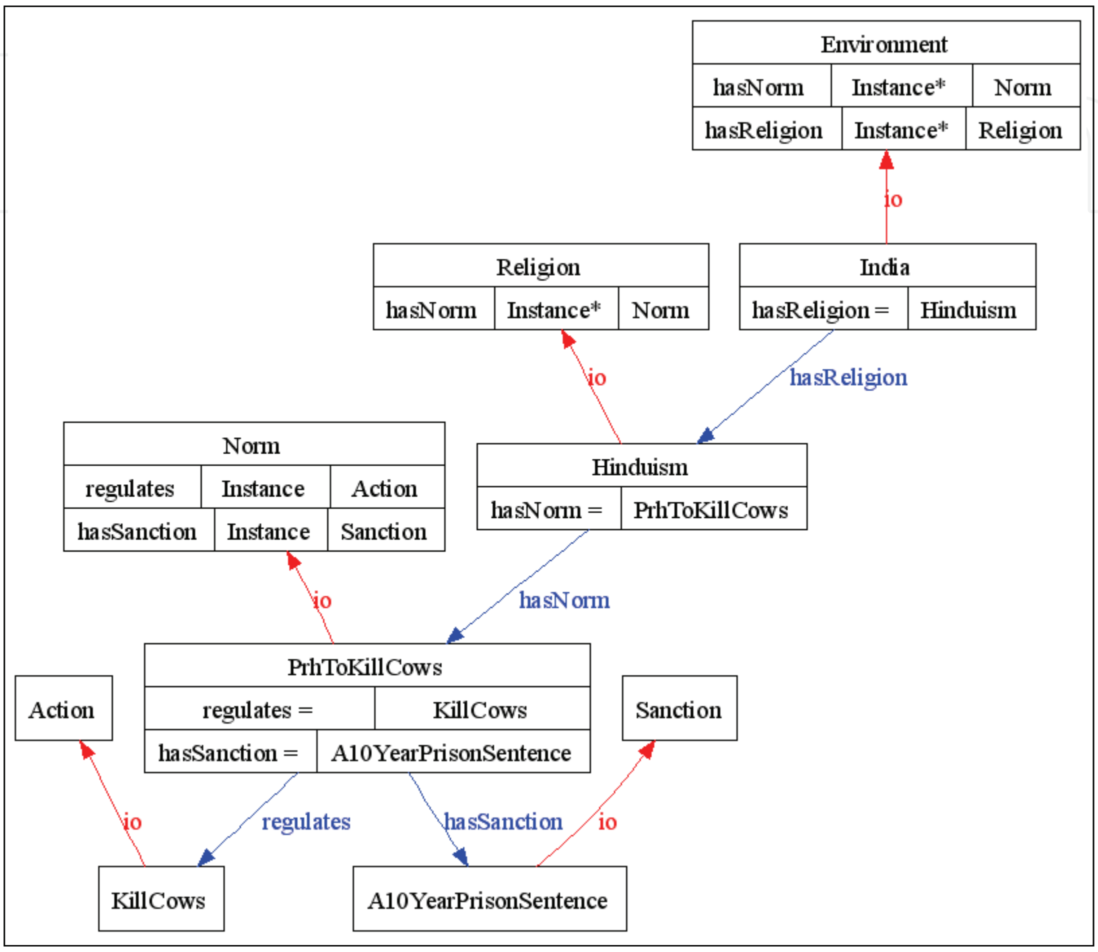


Fig. 11. A Hinduism religious norm concretized in the India environment

3.2 *A Priori* norm enforcement

In order to support *a priori* norm enforcement, DynaCROM has to be enhanced with an enforcer that will be in charge of guaranteeing norm compliance when there is an attempt to violate a norm.

Experiments were made integrating DynaCROM with SCAAR and MOSES, two solutions for norm enforcement. In SCAAR [Chopinaud *et al.*, 2006], the enforcement is done based on the internal behavior of agents; and in MOSES [Minsky_MOSES, URL], it is based on the external behavior of agents. For both solutions, DynaCROM works providing precise norm information as their input.

In the following section, an overview of how DynaCROM works as an input mechanism for norm enforcement solutions is presented. Then, in the two subsequent sections, the norm enforcement based on the agents' external and internal behaviors are explained. Because the enforcement solution is not the focus of DynaCROM, this chapter does not deal with the problems related to that part (*e.g.*, malfunction of the enforcer).

3.3 DynaCROM as an input mechanism for norm enforcement solutions

DynaCROM can be used for providing information as input to norm enforcement solutions. For this, each time an agent starts the execution of a regulated action, DynaCROM retrieves, in the domain ontology, the applicable norms according to the agent's current contexts and, then, sends those norms to be enforced by the chosen norm enforcement solution.

Figure 12 illustrates an overview of the process for contextual norm enforcement. Once an agent executes a regulated action, DynaCROM verifies, in the domain ontology instance, the norms of the action, according to the agent's current contexts. Then, DynaCROM concretizes those norms in a file that is used by the chosen enforcer as its input. The enforcer reads the input file and, then, enforces the norms of the performed action.

In the case of *a posteriori* norm enforcement, the information about the violated norms is sent back to DynaCROM for the application of sanction actions. In this phase, a third-party sanction system can be used for enhancing the DynaCROM solution. However, this idea is not developed in the text of this chapter, but in [Silva, 2008] more information about this issue can be found.

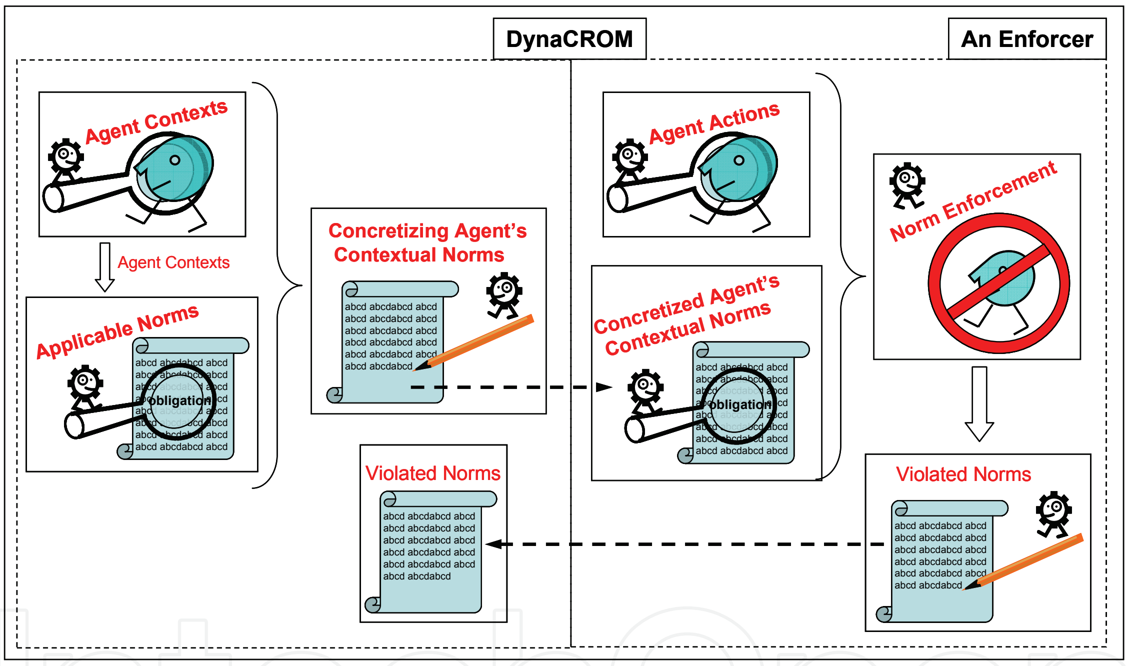


Fig. 12. DynaCROM providing contextual norm information as input to enforcers.

3.4 Contextual norm enforcement based on agents' behavior

Contextual norm enforcement can be done based on the agents' external or internal behavior. In order to exemplify both situations, the following simplification of the FIPA Contract-Net interaction protocol [FIPA_Contract-Net, URL] is considered:

1. A manufacturer wants to build 100 computers;
2. He issues a call for proposal (CFP) to computer suppliers;
3. Computer suppliers answer the CFP with their proposed price;
4. The manufacturer chooses one proposal among the ones he received and informs his decision to the chosen supplier.

In the example, the action of suppliers to 'propose a price' is regulated by the norm for effecting negotiations ("*negotiations are obliged to be paid by using the national currency of the* *seller's country*") and by the norm for calculating prices ("*a state corporate income tax rate is obliged to be imposed on all sales, for immediate delivery or if the deliver address is in North America*"). Both norms were previously mentioned in the text of this chapter. For the enforcement of the last norm, the base price of a computer is predefined to make it possible to calculate its acceptable minimum price.

Norm Enforcement Based on the Agents' External Behavior

An example in which the norms for payments are enforced by a created police agent, based on the agents' external behavior, is illustrated in Figure 13. In brief, the *MissourianManufacturer* (an agent playing the *manufacturer* role in the *Missouri* environment) sends a CFP to the *JapaneseSupplier* (an agent playing the *supplier* role in the *Japan* environment). The *JapaneseSupplier* answers the CFP message with a PROPOSE message in which the currency value is different from the one expected (*JPY* instead of *USD*, the national currency of *USA*).

When the message arrives at the *ManufacturerPolice* (the police agent created to enforce the system norms in the manufacturer agent), the *ManufacturerPolice* blocks the sending of the message to the *MissourianManufacturer* agent and sends an INFORM(*Nok*,*NationalCurrency*) message with the error occurred (*i.e.*, wrong currency) to the *JapaneseSupplier*. Then, the *JapaneseSupplier* sends a new PROPOSE message with the correct currency, however, he does not considered the state corporate income tax of *6.25* from *Missouri*. So, the *ManufacturerPolice* also enforces the other norm and sends an INFORM(*Nok*, *StateCorporateIncomeTaxOf*) message to the *JapaneseSupplier*, informing him that now the error is with the missing state corporate income tax.

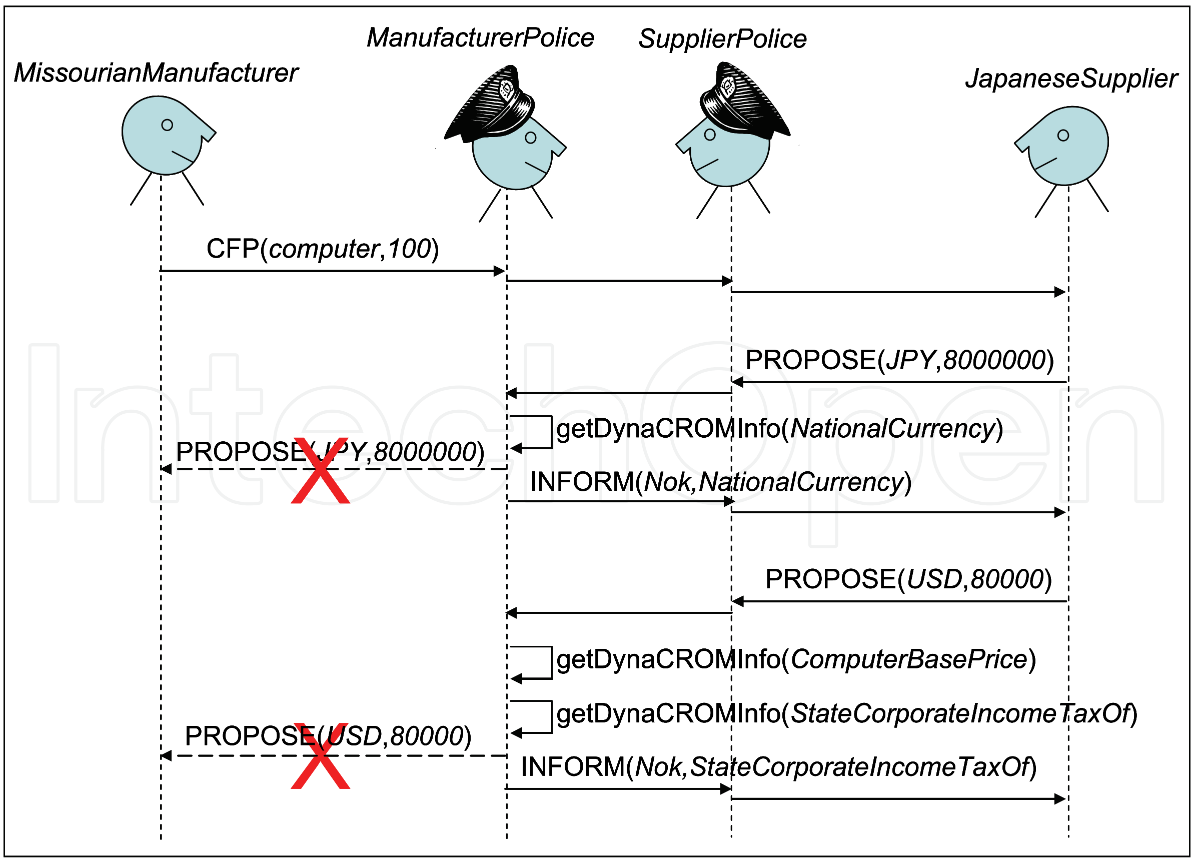

ManufacturerPolice
SupplierPolice
MissourianManufacturer
JapaneseSupplier
CFP(computer,100)
PROPOSE(JPY,8000000)
getDynaCROMInfo(NationalCurrency)
PROPOSE(JPY,8000000)
INFORM(Nok,NationalCurrency)
PROPOSE(USD,80000)
getDynaCROMInfo(ComputerBasePrice)
getDynaCROMInfo(StateCorporateIncomeTaxOf)
PROPOSE(USD,80000)
INFORM(Nok,StateCorporateIncomeTaxOf)

Fig. 13. An example of a police agent enforcing contextual norms for payments

Norm Enforcement Based on the Agents' Internal Behavior

An example in which the norms for payments are enforced by the agents themselves (*i.e.*, agents are self-regulated), based on their internal behavior, is illustrated in Figure 14. In brief, the *MissourianManufacturer* sends a CFP to the *JapaneseSupplier*. The *JapaneseSupplier* tries to answer the CFP message with a PROPOSE message, but, because the proposed currency value is different from the one expected (*JPY* instead of *USD*, the national currency of *USA*), the message is not sent due to the (self-)enforcement of the norm for payments with the national currency.

Then, the supplier agent tries to send a new PROPOSE message (now, with the correct currency), however, he does not consider the state corporate income tax of *6.25* from *Missouri*. Then, the message is not sent due to the (self-)enforcement of the norm for payments with the state corporate income tax.

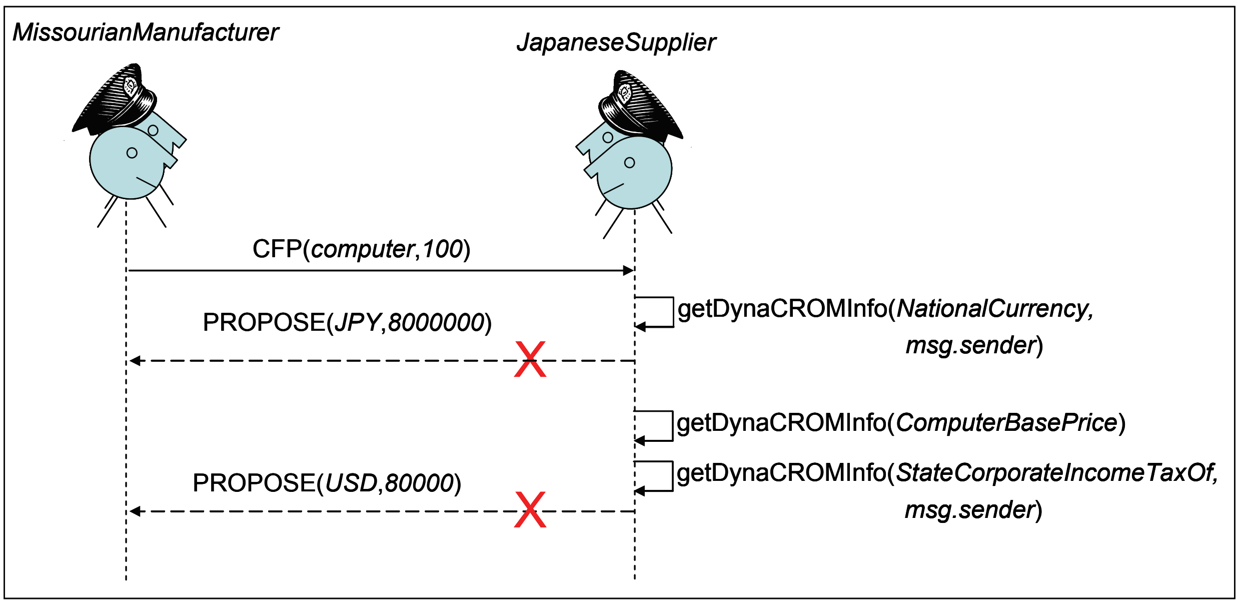

MissourianManufacturer
JapaneseSupplier
CFP(computer,100)
getDynaCROMInfo(NationalCurrency, msg.sender)
PROPOSE(JPY,8000000)
getDynaCROMInfo(ComputerBasePrice)
getDynaCROMInfo(StateCorporateIncomeTaxOf, msg.sender)
PROPOSE(USD,80000)

Fig. 14. Self-regulated agents enforcing contextual norms for payments

4. Related work

Despite all efforts made to move theory and practice of MAS from closed to open agent societies, current solutions do not yet explicitly support openness and its consequences. More precisely, methodologies, modeling languages and tools (*e.g.*, frameworks, platforms), needed for implementing open MAS, do not conveniently cover the aspects of regulation and domain representation for society differentiation.

Traditional modeling of MAS [Bresciani *et al.*, 2004; Cervenka *et al.*, 2005; Odell *et al.*, 2000; Wooldridge *et al.*, 2000] often assumes an individualistic perspective in which agents are considered as autonomous entities that pursue their individual goals, based on their own beliefs and capabilities. Even in this perspective, global behavior emerges from individual interactions and, therefore, the modeling has to be expanded to consider not only an *agent-centric* view, but also *societal* and *organizational-centric* views [Silva *et al.*, 2008]. Furthermore, the overall problem of analyzing the social, legal, economic and technological dimensions of an agent organization must be considered.

Agent-centered approaches can be useful for closed systems, composed of a small number of agents, but they fail to design open systems [Rodríguez-Aguliar, 2001; Esteva, 2003]. For instance, in critical applications such as those within business, environments or government agencies (hospitals, police, justice, etc.), the structural characteristics of the domain have to be incorporated. That is, the design of an agent society must also consider organizational characteristics such as stability over time, some level of predictability, commitment to aims and strategies, and so on.

The idea of modeling MAS as organizations was early proposed by [Gasser *et al.*, 1987; Pattison *et al.*, 1987; Corkill and Lesser, 1983; Werner, 1987] and it is still a major issue in the MAS research field, especially in applications on the areas of Service Oriented Computing, Grid Computing and Ambient Intelligence. Recently, the subject of MAS design from the organizational perspective has been mainly discussed in the COIN workshop (meaning workshop on *Coordination, Organization, Institutions and Norms* in agent systems) [COIN, URL].

The COIN workshop series started in 2005 during the ANIREM [Lindemann *et al.*, 2005] and OOOP [Boissier *et al.*, 2005] workshops held in AAMAS'05 [Kraus and Singh, 2005]. The series has been held yearly since then, as a dual event co-located within large international conferences of the area in different geographic regions (*e.g.*, in 2008, at AAAI'08 [Dignum and Matson, 2008] in the USA and at AAMAS'08 [Hübner and Boissier, 2008] in Portugal; in 2007, at AAMAS'07 [Ossowski and Sichman, 2007] in Hawaii and at MALLOW'07 [Noriega and Padget, 2007] in UK; in 2006, at ECAI'06 [Boella *et al.*, 2006a] in Italy and at AAMAS'06 [Dignum *et al.*, 2006] in Japan).

Even with this research effort, organizational approaches have not been a common use in MAS, which is usually seen as a pure aggregation of agents. The fact that organizational approaches have not been effectively adopted suggests that some work still needs to be done in providing better tools for the design and implementation of MAS. System developers need to be supported when dealing with MAS in which intrinsic characteristics of the application domain (*e.g.*, society structure) have to be considered. This necessity increases when considering open systems from particular 'cultures'[7].

According to [Jennings, 2001] there are two points that qualitatively differentiate agent interactions from those that occur in other software engineering paradigms. First, agent-oriented interactions generally occur through a high-level (declarative) agent communication language, which is often based on the speech act theory [Mayfield *et al.*, 1995]. Secondly, agents need the computational apparatus to make context-dependent decisions about the nature and scope of their interactions and to initiate (and respond to) interactions that were not initially foreseen.

Regarding these distinctions, an appropriate solution for regulating interactions among agents cannot be rigidly fixed at any system phase and should continuously support data updates according to the changing contexts of agents. Moreover, besides hierarchical relationships among participants in interrelated subsystems, non-hierarchical relationships is also relevant information for norm enforcement in MAS.

Thus, it makes it necessary to provide a contextual normative solution in which different types of relationships among agents can be dealt with in order to enable norm enforcement

[7] 'Culture' meaning "the predominating attitudes and behavior that characterize the functioning of a group or organization" [OnLineDictionary, URL].

in MAS. The solution should be flexible enough for supporting norm evolution and it should not be only based on the interaction level, but also on others domain levels.

5. Conclusion

Three main assumptions underlie this research. Firstly, MAS has emerged as a concrete solution to develop complex software systems in which monolithic architectures (based on objects) have been replaced by distributed ones (based on agents). Secondly, with the advent of the Semantic Web and its technologies (*e.g.*, new ontologies' languages as OWL), agents will be able to process information from different sources. In this way, they will be able to move around other MAS looking for resources and/or services not found locally. In this scenario, openness will be an intrinsic and mandatory characteristic of upcoming systems. However, openness without control leads to chaotic scenarios. The use of norms in MAS is a promising approach for achieving openness in a reliable way. So, the final assumption of this work is that MAS should be normative.

In this chapter, the theoretical fundamentals of the DynaCROM methodology developed to support the system developer in the tasks of implementation, management and evolution of the norms of his MAS is presented. The methodology includes the phases of: (i) contextualization, through a top-down classification for contextual norms; (ii) concretization and representation, through a contextual normative ontology; and, finally, (iii) composition of norms, through a norm composition process.

The top-down classification for contextual norms proposed by DynaCROM facilitates the tasks of elicitation, organization and management of norms. The DynaCROM contextual normative ontology supports heterogeneous agents with a common understanding about the system norms. The norm composition process defined by DynaCROM makes it easy to update system regulation by both evolving norms in a unique resource (an ontology) and/or by customizing particular rules for different compositions of contextual norms.

Although application agents can be informed about their current (contextual) norms, by using the DynaCROM behavior, agents' developers can implement their agents regardless of this information. In this case, agents need a solution that continuously informs them about system data, according to their current contexts, in order to deal with the applicable norms of each action performed by them.

The subject of contextual norm enforcement is also analyzed in this chapter. The achievement of a norm enforcement contextualized for each application agent is due to the integration of DynaCROM with third-party enforcers. The integration of MOSES with DynaCROM permits a contextual norm enforcement based on the agent's external behavior. The integration of SCAAR with DynaCROM permits a contextual norm enforcement based on the agent's internal behavior. For both cases, norm enforcement was done *a priori*. Nevertheless, *a posteriori* norm enforcement is presented in the beginning of the chapter, where an overview of a DynaCROM solution for it is also given.

As future work, DynaCROM should encompass a formal method amenable to rigorous verification of the system developer's specifications. In the current solution, *formal* versus *non-formal* issues in DynaCROM are restricted to the available properties of the chosen ontology and rule languages for norm representation and composition, respectively.

DynaCROM should also propose how to deal with constitutive, procedural and conditional norms. New examples of how those types of norms should be given to guide system developers interested in that solution. Other interesting future works are to consider time restrictions in DynaCROM, and suggest solutions for conflicting norms from the same or different levels of abstractions.

6. References

Alberti, M.; et al. Mapping Deontic Operators to Abductive Expectations. *Journal of Comput. Math. Organ*. Theory, v.12, Issue 2–3, p.205–225, 2006.

Artikis, A.; et al. Animated specifications of computational societies. *1st International Joint Conference on Autonomous Agents and Multiagent Systems*, ACM, Italy, p.1053–1061, 2002.

Baader, F.; et al. The Description Logic Handbook: Theory, Implementation, Applications. Cambridge University Press, UK, 2003. ISBN 0-521-78176-0.

Bechhofer, S.; et al. OWL Web Ontology Language Reference. Recommendation from W3C since February, 2004. URL: <http://www.w3.org/TR/owl-ref/>.

Boella, G.; et al. *Workshop on Coordination, Organization, Institutions and Norms at the 17th European Conference on Artificial Intelligence*, (COIN@ECAI2006), 2006a.

Boella, G.; et al. Introduction to normative multiagent systems. *Journal of Comput. Math. Organ. Theory*. v.12, Issue 2–3, p.71–79, 2006b, ISSN 1381-298X.

Boissier, O.; et al. International Workshop on Organizations in Multi-Agent Systems FROM ORGANIZATIONS TO ORGANIZATION ORIENTED PROGRAMMING IN MAS at the *4th International Conference on Autonomous Agents and Multiagent Systems* (OOOP@AAMAS2005), July, 2005, the Netherlands.

Bouquet, P.; et al. C-OWL: Contextualizing Ontologies. In: 2nd International Semantic Web Conference (ISWC-03), *Lecture Notes on Computer Science*, 2870, p.164–179, 2003.

Breitman, K.K.; et al. Using Ontologies to Formalize Services Specifications. In: *Multi-Agent Systems*, 3rd NASA - Goddard/ IEEE Workshop FAABS III, Formal Approaches to Agent-Based Systems. Greenbelt, MA. 2004.

Bresciani, P.; et al. Tropos: an agent-oriented software development methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, v.8, n.3, p.203–236, 2004.

Castelfranchi, C.; et al. Deliberative Normative Agents: Principles and Architecture. In: *Agent Theories, Architectures, and Languages* (ATAL-99), p.364–378, 1999.

Cervenka, R.; et al. AML: Agent Modeling Language toward industry-grade agent-based modeling. *Agent-Oriented Software Engineering* V, Lecture Notes in Computer Science, v.3382, p.31–46, 2005.

Chopinaud, C.; et al. Prevention of harmful behaviors within cognitive and autonomous agents. *7th European Conference on Artificial Intelligence* (ECAI'06), 205–209, 2006.

COIN. Website of the Coordination, Organization, Institutions and Norms in agent systems workshop. URL: <http://www.pcs.usp.br/~coin/>.

Dey, A. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1), p.4–7, 2001, ISSN: 1617-4909.

Dignum, F. Abstract Norms and Electronic Institutions. In: *International Workshop on Regulated Agent-Based Social Systems: Theories and Applications* (RASTA'02), p.93–104. Bologna, Italy, 2002.

Dignum, V.; et al. Workshop on Coordination, Organization, Institutions and Norms at the *5th International Conference on Autonomous Agents and Multiagent Systems* (COIN@AAMAS2006), May 09, 2006, Japan.

Dignum, V. and Matson, E. Workshop on Coordination, Organization, Institutions and Norms at the *23rd Conference on Artificial Intelligence* (COIN@ AAAI2008), July 13-14, 2008, Chicago, USA.

Esteva, M. Electronic Institutions: from specification to development. N. 19 in the Monograph Series of the Institut d'Investigació en Intelligència Artificial (IIIA), PhD thesis, Technical University of Catalonia, 2003.

Esteva, M.; et al. AMELI: An Agent-based Middleware for Electronic Institutions. In: *3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, p.236–243, 2004.

FACT. Website of the Fast Classification of Terminologies. URL: <http://www.cs.man.ac.uk/~horrocks/FaCT/>.

Felicíssimo, C. H.; et al. Normative Ontologies to Define Regulations Over Roles in Open Multi-Agent Systems. *AAAI Fall Symposium: Roles, an Interdisciplinary Perspective: Ontologies, Programming Languages, and Multiagent Systems*. AAAI Press. v.1. p.68-72, 2005b.

Felicíssimo, C.; et al. Providing Contextual Norm Information in Open Multi-Agent Systems. *Post Proceedings of the 8th International Bi-Conference Workshop on AGENT-ORIENTED INFORMATION SYSTEMS* IV, Lecture Notes in Computer Science, v.4898, p.19–36, 2008a, ISBN: 978-3-540-77989-6.

Felicíssimo, C. H.; et al. Contextualizing Normative Open Multi-Agent Systems. In: *23rd Annual ACM Symposium on Applied Computing* (ACM SAC 2008), Fortaleza, Brazil, March 16-20, 2008b.

Felicíssimo, C.; et al. DynaCROM: An Approach to Implement Regulative Norms in Normative Multiagent Systems. In: *3th DEON'08 International Workshop on Normative Multiagent Systems of the 9th International Conference on Deontic Logic in Computer Science* (NorMAS@DEON'08). Luxembourg, 2008d.

Felicíssimo, C.; et al. How to Concretize Norms in NMAS? An Operational Normative Approach Presented with a Case Study from the Television Domain. In: *AAAI'08 International Workshop on Coordination, Organization, Institutions and Norms in agent systems* (COIN@AAAI'08). USA, 2008e.

Ferber, J.; et al. From Agents to Organizations: an Organizational View of Multi-Agent Systems. In: *Agent-Oriented Software Engineering* IV (AOSE, 2003), Lecture Notes in Computer Science, v.2935, p.214–230, 2003.

FIPA_Contract-Net. Website of the FIPA Contract Net Interaction Protocol Specification. URL: <http://www.fipa.org/specs/fipa00029/SC00029H.html>.

Gaertner, D.; et al. Distributed Norm Management in Regulated Multi-Agent Systems. In: *6th International Joint Conference on Autonomous Agents and Multiagent Systems*, p.624–631, 2007.

Gasser, L.; et al. Distributed Artificial Intelligence, chapter MACE: A flexible test-bed for distributed *AI research*, p.119–152, Pitman Publishers, 1987.

Grizard, A.; et al. A peer-to-peer normative system to achieve social order. In: *AAMAS'06 Workshop on Coordination, Organization, Institutions and Norms in agent systems* (COIN@AAMAS-2006), 2006, Japan.

Grossi, D. and Dignum, F. From Abstract to Concrete Norms in Agent Institutions. 3rd NASA Workshop on Formal Approaches to Agent-Based Systems (FAABS III). *Lecture Notes in Computer Science*, 3228, 12–29, 2004, ISBN: 978-3-540-24422-6.

Grossi, D.; et al. Classificatory Aspects of Counts-as: An Analysis in Modal Logic. In: *Journal of Logic and Computation*, v.16, n.5, p.613–643, 2006.

Gruber, T. R. A translation approach to portable ontology specifications. In: *Knowledge Acquisition*, 5(2), p.199–220, 1993, ISSN: 1042-8143.

Hewitt, C. Open Information Systems Semantics for Distributed Artificial Intelligence. *Artificial Intelligence*, v.47(1-3), p.79–106,1991, ISSN:0004-3702.

Hübner, J. F.; et al. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In: Advances in Artificial Intelligence: 16th Brazilian Symposium on Artificial Intelligence (SBIA 2002). Brazil, 2002. *Lecture Notes in Artificial Intelligenc*, v.2507. p.118–128, Berlin, Springer. 2002.

Hübner, J.F. and Boissier, O. Workshop on Coordination, Organization, Institutions and Norms at the *7th International Conference on Autonomous Agents and Multiagent Systems* (COIN@AAMAS2008), May 12-13, 2008, Portugal.

JENA. Website of Jena – A Semantic Web Framework for JAVA. URL: <http://jena.sourceforge.net/>.

Jennings, N.; et al. A Roadmap of Agent Research and Development. *Journal of Agents and Multi-Agent Systems*, v.1, p.7–38, 1998.

Jennings, N. R. On Agent-Based Software Engineering. *Artificial Intelligence*, 117(2), p.277–296, 2000.

Jennings, N. R. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4), p.35–41, 2001.

Jones, A.J.I. and Sergot, M. On the characterization of law and computer systems: the normative systems perspective. *Deontic Logic in Computer Science*, 1993.

Kagal, L. and Finin, T. Modeling conversation policies using permissions and obligations. *Journal of Agents and Multi-Agent Systems*, v.14, p.187–206, 2007.

Khedr, M. and Karmouch, A. ACAI: Agent-Based Context-aware Infrastructure for Spontaneous Applications. *Journal of Network & Computer Applications*, 28(1), p.19–44, 1995.

Kraus, S. and Singh, M. In: *4th International Conference on Autonomous Agents and Multiagent Systems* (AAMAS2005), July, 2005, the Netherlands.

Lindemann, G.; et al. Agents, Norms and Institutions for Regulated Multiagent Systems at the *4th International Conference on Autonomous Agents and Multiagent Systems* (ANIREM@AAMAS2005), July, 2005, the Netherlands.

Mayfield, J.; et al. Evaluating KQML as an agent communication language. *Intelligent Agents* II, p.347–360, 1995.

Minsky_LGI. Website of the Law Governed Interaction (LGI): A Distributed Coordination and Control Mechanism (An Introduction and a Reference Manual), URL: <http://www.cs.rutgers.edu/~minsky/papers/manual.pdf>.

Minsky_MOSES. The MOSES toolkit. URL:<http://www.moses.rutgers.edu/>.

Noriega, P. and Padget, J. In: Workshop on Coordination, Organization, Institutions and Norms at the Multi-Agent Logics, Languages, and Organisations — *Federated Workshops* (COIN@MALLOW2007), p.3–4, 2007, Uk.

Noy, N. and Rector, A. (eds.): Defining N-ary Relations on the Semantic Web: Use with Individuals, 2007, URL: <http://www.w3.org/TR/swbp-n-aryRelations>.

Odell, J.; et al. Extending UML for agents. In: *Agent-Oriented Information Systems Workshop at the 17th National Conference on Artificial Intelligence*, p.3–17, 2000.

Ossowski, S. and Sichman, J.S. *Workshop on Coordination, Organization, Institutions and Norms at the 6th International Conference on Autonomous Agents and Multiagent Systems*, May 14-18, 2007, Honolulu, Hawaii, USA.

PELLET. Open Source OWL DL Reasoner. URL: <http://pellet. owldl.com/>.

RACER. Website of the Renamed Abox and Concept Expression Reasoner software. URL: <http://www.sts.tu-harburg.de/~r.f.moeller/racer/>.

Rodríguez-Aguilar, J.A. On the Design and Construction of Agent-mediated Electronic Institutions, PhD thesis, Universitat Autonoma de Barcelona, 2001.

Schilit, B. and Theimer, M. Disseminating Active Map Information to Mobile Hosts. In: *IEEE Network*, 8(5), p.22–32, 1994.

Silva, V. T. da and de Lucena, C.J.P. From a conceptual framework for agents and objects to a multi-agent system modeling language. *Journal of Autonomous Agents and Multi-Agent Systems*, v.9(1-2), p.145–189, 2004b, ISSN 1387-2532.

Silva, V. T. da. From the specification to the implementation of norms: an automatic approach to generate rules from norms to govern the behavior of agents. In: *Journal of Autonomous Agents and Multi-Agent Systems*, v.17(1), p.113–155, 2008, ISSN: 1387-2532.

Silva, V. T. da; et al. MAS-ML: a multiagent system modelling language. In: *International Journal of Agent-Oriented Software Engineering*, v.2(4), p.382–421, 2008.

Simon, H. A. The Sciences of the Artificial. MIT Press. 1996.

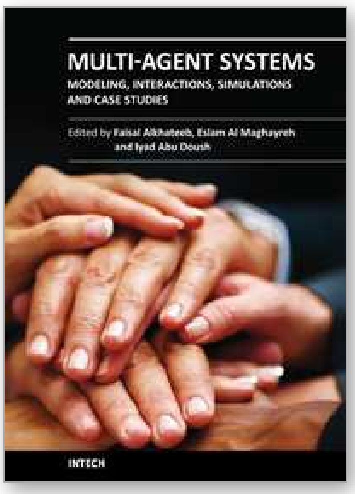
Thomas, G. and Williams, A. B. Roles in the Context of Multiagent Task Relationships. AAAI Fall Symposium "Roles, an Interdisciplinary Perspective: Ontologies, Programming Languages, and Multiagent Systems", ISBN: 978-1-57735-254-9, 2005.

Vázquez-Salceda, J.; et al. Organizing Multiagent Systems. In: *Journal of Autonomous Agents and Multi-Agent Systems*, 11(3), p.307–360, 2005.

Weyns, D.; et al. Environment as a first class abstraction in multiagent systems. In: *Journal of Autonomous Agents and Multi-Agent Systems* (JAAMAS, 07), v.14(1), p.5–30, 2007, ISSN: 1387-2532.

Wooldridge, M.; et al. The Gaia Methodology for Agent- Oriented Analysis and Design. In: *Journal of Autonomous Agents and Multi-Agent Systems*, v.3(3), p.285–312, 2000.

Wright, G.H.v. Deontic Logic. In: *Mind New Series*, v.60(237), p.1–15, 1951.


MULTI-AGENT SYSTEMS
MODELING, INTERACTIONS, SIMULATIONS AND CASE STUDIES
Edited by Faisal Alkhateeb, Eslam Al Maghayreh and Iyad Abu Doush
INTECH

Multi-Agent Systems - Modeling, Interactions, Simulations and Case Studies
Edited by Dr. Faisal Alkhateeb

ISBN 978-953-307-176-3
Hard cover, 502 pages
Publisher InTech
Published online 01, April, 2011
Published in print edition April, 2011

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821