Providing Contextual Norm Information in Open Multi-Agent Systems

Carolina Felicíssimo[1, 3], Ricardo Choren[2], Jean-Pierre Briot[1, 3], Carlos Lucena[1], Caroline Chopinaud[3] and Amal El Fallah Seghrouchni[3]

[1] DI, PUC-Rio: Rua M. de São Vicente, 225, Gávea Rio de Janeiro, RJ, 22453-900, Brasil
{cfelicissimo,lucena}@inf.puc-rio.br
[2] SE-8, IME: Pca General Tiburcio 80, 22290-270, Rio de Janeiro RJ, Brazil
choren@de9.ime.eb.br
[3] LIP6, Paris VI: 104 Avenue Kennedy, 75016, Paris, France
{jean-pierre.briot,caroline.chopinaud,amal.elfallah}@lip6.fr

Abstract. Agents can freely migrate among open MASs in order to obtain resources or services not found locally. In this scenario, agent actions should be guided for avoiding unexpected behavior. However, open MASs are extremely dynamic and, thus, a solution for guiding agent actions is non-trivial. This work details DynaCROM, our solution for continuously supporting agents in open MASs with updated norm information. The main asset of DynaCROM is that it decreases the complexity of norm representation by using contexts. DynaCROM proposes (i) a top-down modeling of contextual norms, (ii) an ontology to explicitly represent norm semantics and (iii) a rule inference engine to customize different compositions of contextual norms. Thus, DynaCROM offers a solution for both developers to enhance their open MASs with norm information and agents to be continuously supported with precise norm information.

1 Introduction

Multi-agent systems (MASs) have emerged as a promising approach to develop information systems that clearly require several goal-oriented problem-solving entities [21]. Following this direction, we believe that near information systems will be implemented as open MASs, which will be composed of many sets of heterogeneous self-interested agents. These agents will be mobile agents, i.e. they will have the capability to freely migrate among MASs for obtaining resources or services not found locally. An MAS can be considered an open system when it presents the following characteristics [13]:

Heterogeneity: agents are possibly developed by different parties, in different programming languages, with different purposes and preferences.

Accountability: agent actions must be monitored to detect the execution of behaviors that may not be according to the overall expected functioning of the system.

Social change: agent societies are not static; they may evolve over time by updating their information. So, future changes should be easily accommodated.

In open MASs, agents may be heterogeneous, but they all must know how to assimilate provided information for effective execution. In this sense, information should be expressed in a meaningful way for agents, avoiding misunderstandings. Moreover, the intrinsic dynamics of open MASs should be supported by a flexible mechanism that easily permits data updates. Regarding these points, we developed a solution for continuously providing contextual norm information to agents in open MASs. In our opinion, open MASs should be enhanced with norm information for guiding agent actions. Our solution, called DynaCROM (meaning *dynamic contextual regulation information provision in open MASs*)[8-11], proposes (i) a top-down modeling of contextual norms, (ii) an ontology to explicitly represent norm semantics and (iii) a rule inference engine to customize different compositions of contextual norms.

DynaCROM allows for accountability since MASs and their agents continuously have information about which norms they should follow. DynaCROM also allows for social changes since information is defined in a central resource (an ontology) that can easily have its data composed (thanks to a rule support) and updated.

It is important to stress here that, in this work, we make no assumptions about whether agents decide or not to be compliant with norms. DynaCROM allows the modeling and representation of customized compositions of contextual norms, offering precise norm information for agents to consider in a given context. Thus, DynaCROM provides the way for agents to reason about norm compliance and for developers to implement normative MASs. Norm-aware agents are more likely to perform correctly and, consequently, to achieve their goals faster.

The remainder of this paper is organized as follows. Section 2 details DynaCROM, including its top-down modeling, use of ontology and rules, and implementation. Section 3 describes a case study. Section 4 briefly presents how DynaCROM answers (contextual norm information) can be used. Section 5 compares DynaCROM with two related works. Finally, Section 6 concludes the paper and outlines future works.

2 Contextual Norm Information Provision in Open MASs

MASs are generally made up of environments, organizations and agents [20]. Environments [36] are discrete computational locations (similar to places in the physical world) that provide conditions for agents to inhabit it. Environments can have refinement levels, such as a specialization relationship (e.g., country, state, etc.), but there cannot be overlaps (e.g., there cannot be two countries in the same place). An environment also can have many organizations. Organizations [12] are social locations inside which groups of agents play roles. Furthermore, an organization can have many sub-organizations, but each organization belongs to only one environment [28]. An agent can be in different organizations; and, agents with the mobility characteristic can migrate among environments or organizations. Roles [32] are abstractions that define a set of related tasks for agents achieving their designed goals. Agents interact with other agents from the same or different environments, organizations and roles. Environments, organizations, roles and agent interactions suggest different contexts (*implicit situational information* [7]) found in open MASs.

2.1 Modeling Contextual Norms

Research in context-aware applications suggests top-down architectures for modeling contextual information [22,18]. Thus, DynaCROM defines that norms of MASs should be modeled according to the following four levels of abstractions: *Environment*, *Organization*, *Role* and *Interaction* contexts. We call these contexts *regulatory contexts* and they are differentiated by the boundaries of their data (norms). *Environment Norms* are applied to all agents in a regulated environment; *Organization Norms* are applied to all agents in a regulated organization; *Role Norms* are applied to all agents playing a regulated role; *Interaction Norms* are applied to all agents involved in a regulated interaction.

Fig. 1 illustrates the boundaries of environment, organization, role and interaction norms. There, agents are regulated by compositions of contextual norms. For instance, the agents on the left side of the figure are regulated by compositions of common environment and interaction norms and by compositions of different organization and role norms; the agents on the right side of the figure are regulated by compositions of common environment, organization and interaction norms and by compositions of different role norms; the two agents interacting, from the different environments, are regulated by compositions of different environment, organization, role and interaction norms.

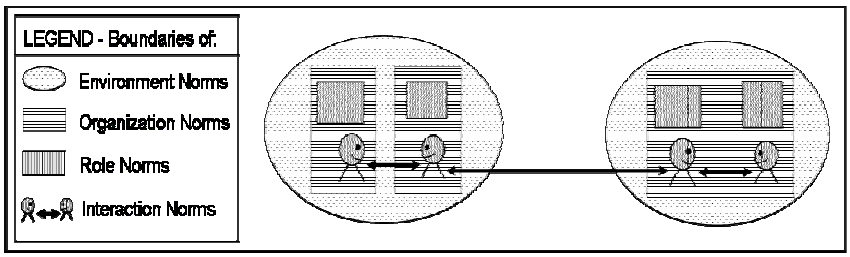

Fig. 1. The boundaries of *Environment*, *Organization*, *Role* and *Interaction Norms*

We believe that the four DynaCROM regulatory contexts are not targeted to a particular application domain, but rather they represent a minimum set for a general contextual norm information provision in open MASs. For more complex MASs, this set should be improved with additions and refinements of regulatory contexts for representing particular domain norms.

Norms define which actions are *permitted*, *obliged* and *prohibited* to be executed by agents in an open MAS. A *permitted norm* defines that an act is allowed to be performed; an *obliged norm* defines that an act must be performed; a *prohibited norm* defines that an act must not be performed. These three types of norms described represent the three fundamental deontic statuses of an act [1] from Deontic Logic [37]. Deontic Logic makes it possible to address the issue of explicitly and formally defining norms and dealing with the possibility of violation. In normative (i.e., regulated) open MASs, agents need to be norm-aware entities for taking into account the existence of social norms in their decisions (either to follow them or to violate them) and to react to norm violations by other agents [5].

2.2 Representing Contextual Norms

Norms should have their semantics explicitly expressed in a meaningful way for heterogeneous agents to process their contents. Regarding this, DynaCROM uses ontologies for representing its regulatory contexts and data. For the DynaCROM ontology, the following definitions are valid: an *ontology* is a conceptual model that embodies shared conceptualizations of a given domain [17]; a *contextual ontology* is an ontology that represents contextual information [3]; and, a *contextual normative ontology* is an ontology that represents contextual norm information, having the norm concept as a central asset. The DynaCROM contextual normative ontology, or simply the DynaCROM ontology, is illustrated in Fig. 2.

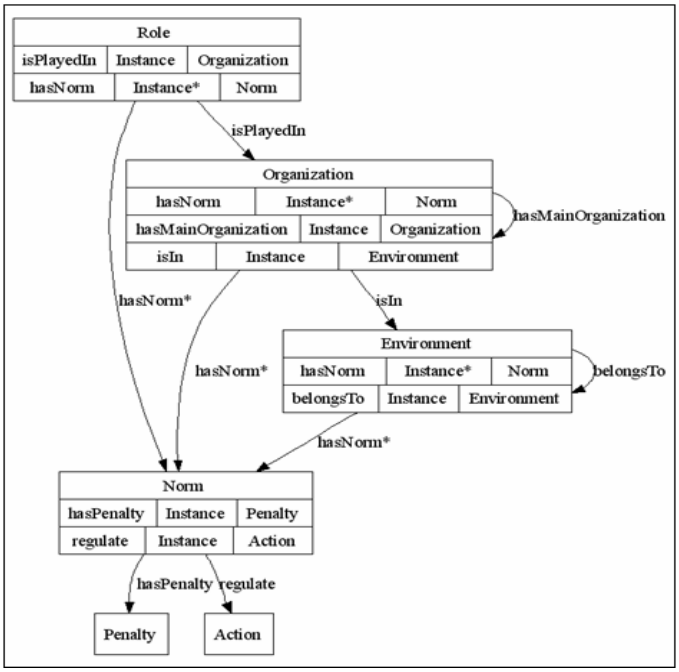

Fig. 2. The DynaCROM ontology

The DynaCROM ontology defines six related concepts (see Fig. 2), all at the same hierarchical level, for representing its environment, organization and role regulatory contexts. The *Action* concept encompasses all instances of regulated actions. The *Penalty* concept encompasses all instances of fines to be applied when norms are not fulfilled. The *Norm* concept encompasses all instances of norms from all regulatory contexts. The *Environment* concept encompasses all instances of regulated environments; and, each environment encompasses its associated norms and its owner environment (the environment it belongs to). The *Organization* concept encompasses all instances of regulated organizations; and, each organization encompasses its associated norms, main organization (the organization to which it is associated) and envi-

ronment. The *Role* concept encompasses all instances of regulated roles; and, each role encompasses its associated norms and organization. The Norm and Penalty concepts are specialized into sub-concepts according to the permitted, obliged and prohibited statuses of an act from Deontic Logic.

In order to effectively be used in an open MASs, the DynaCROM ontology should be instantiated and it probably should be extended with both particular domain concepts and interaction norms. In the DynaCROM ontology, the interaction regulatory context should be implemented by following the representation pattern [26] from a Semantic Web Best Practices document. This pattern defines that the relation object itself must be represented by a created concept that links the other concepts from the relation (i.e. reification of relationship). Thus, in the DynaCROM ontology, an interaction norm should be represented by a new *Norm* sub-concept linking two *Role* concepts. For instance, suppose that a supplier deals with a customer and the interaction between them is regulated by a norm describing the obligation to pay when a deal is done. The interaction norm in the DynaCROM ontology is represented by a new obligation concept, called for example "*ObligationToPay*", linking the supplier and customer *Role* sub-concepts.

2.3 Composing Contextual Norms

After manually classifying and organizing user defined norms, according to a top-down modeling, and explicitly representing these norms into an ontology instance, DynaCROM uses rules to automatically compose contextual norms. This process is simple and it can be summarized as follows: DynaCROM reads an ontology instance for getting data and the information about how concepts are structured; then, it reads a rule file for getting the information about how concepts have to be composed according to activated rules; and, finally, it infers a new ontology instance based on the previous readings. Fig. 3 illustrates an overview of the DynaCROM process.

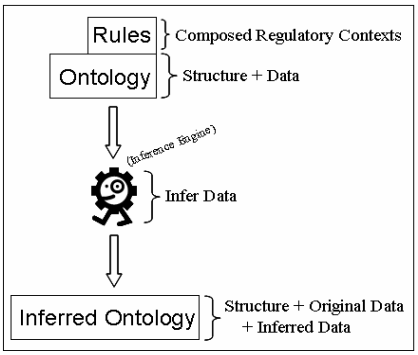

Fig. 3. The DynaCROM process

DynaCROM rules are *ontology-driven* rules, i.e. they are created according to the ontology structure and they are limited according to the number of related concepts to which each concept is linked. DynaCROM has four pre-defined rules for hierarch its

regulatory contexts (e.g., every role has its norms composed with the norms of its organization). These rules, presented in Table 1, receive as input parameters instances of the *Environment*, *Organization* and *Role* concepts from a DynaCROM ontology.

`Rule1` (lines 1 – 4) states that a given environment will have its norms composed with the norms of its owner environment (the environment it is linked to by the "*belongsTo*" relationship). More precisely, the following process is executed: in (4), the owner environment ("`?OEnv`") of the given environment ("`?Env`") is discovered; in (3), the norms of the owner environment ("`?OEnvNorms`") are discovered; finally, in (2), the norms of the owner environment are composed with the norms of the given environment.

`Rule2` (lines 5 – 8) states that a given organization will have its norms composed with the norms of its main organization (the organization it is linked to by the "*hasMainOrganization*" relationship). More precisely, the following process is executed: in (8), the main organization ("`?MOrg`") of the given organization ("`?Org`") is discovered; in (7), the norms of the main organization ("`?MOrgNorms`") are discovered; finally, in (6), the norms of the main organization are composed with the norms of the given organization.

`Rule3` (lines 9 – 12) states that a given organization will have its norms composed with the norms of its environment (the environment it is linked to by the "*isIn*" relationship). More precisely, the following process is executed: in (12), the environment ("`?OrgEnv`") of the given organization ("`?Org`") is discovered; in (11), the norms of the environment ("`?OrgEnvNorms`") are discovered; finally, in (10), the norms of the environment are composed with the norms of the given organization.

`Rule4` (lines 13 – 16) states that a given role will have its norms composed with the norms of its organization (the organization it is linked to by the "*isPlayedIn*" relationship). More precisely, the following process is executed: in (16), it is discovered the organization ("`?Org`") of the given role ("`?Role`"); in (15), the norms of the organization ("`?OrgNorms`") are discovered; finally, in (14), the norms of the organization are composed with the norms of the given role.

Table 1. Rules for hierarch the DynaCROM regulatory contexts by composing their norms

```
(1)  Rule1- [ruleForEnvWithOEnvNorms:
(2)          hasNorm(?Env,?OEnvNorms)
(3)            <- hasNorm(?OEnv,?OEnvNorms),
(4)               belongsTo(?Env,?OEnv)]

(5)  Rule2- [ruleForOrgWithMOrgNorms:
(6)          hasNorm(?Org,?MOrgNorms)
(7)            <- hasNorm(?MOrg,?MOrgNorms),
(8)               hasMainOrganization(?Org,?MOrg)]

(9)  Rule3- [ruleForOrgWithEnvNorms:
(10)         hasNorm(?Org,?OrgEnvNorms)
(11)           <- hasNorm(?OrgEnv,?OrgEnvNorms),
(12)              isIn(?Org,?OrgEnv)]

(13) Rule4- [ruleForRoleWithOrgNorms:
(14)         hasNorm(?Role,?OrgNorms)
(15)           <- hasNorm(?Org,?OrgNorms),
(16)              isPlayedIn(?Role,?Org)]
```

Rules can compose data (e.g., norms) of concepts from the same type (e.g., `Rule1`) or from different types (e.g., `Rule3`), and they also can compose data of concepts directly related (hierarchical form) or indirectly related (non-hierarchical form). Table 2 presents `Rule5`, which is an example of rule for composing the norms of concepts indirectly related (the *Role* and *Environment* concepts from the DynaCROM ontology).

Table 2. A rule for composing the norms of two concepts indirectly related

```
(17) Rule5- [ruleForRoleWithOrgEnvNorms:
(18)          hasNorm(?Role,?OrgEnvNorms)
(19)            <- hasNorm(?OrgEnv,?OrgEnvNorms),
(20)               isIn(?Org,?OrgEnv),
(21)               isPlayedIn(?Role,?Org)]
```

2.4 The DynaCROM Implementation

In open systems, no centralized control is feasible. Their key characteristics are: agent heterogeneity, conflicting individual goals and limited trust [1]. Heterogeneity and autonomy rule out any assumption concerning the way agents are implemented and behave. Thus, a mechanism not hard coded inside agents' original codes and whose data (e.g., norms) can be dynamically updated is the only viable solution for regulations in open MASs [16]. Regarding this, the DynaCROM execution process (see Fig. 3) was implemented as a self-contained JAVA [15] solution and encapsulated as a JADE [33] behavior. Thus, DynaCROM is a general solution that can be used in many application domains without the need of extra implementations. It is only necessary to instantiate the DynaCROM ontology and, probably, to extend it with domain concepts. Domain rules can also be joined with DynaCROM rules.

Table 3 presents the core of the DynaCROM implementation. The process starts when the "*getOntModel()*" method (see line 7) retrieves a DynaCROM ontology instance. This ontology instance represents the regulatory contexts (by the ontology structure) and user defined norms (by the ontology data) from an application domain. The customized compositions of contextual norms are specified by the rules defined in the "*rulesToComposeNorms.rules*" file (called in line 4). The "*reasoner*" variable (see line 5) represents the rule-based inference engine which, based on the retrieved ontology instance and active rules, automatically deduces the customized compositions of contextual norms. This result is kept in the "*inferredModel*" variable (see line 7), which will be continuously read by DynaCROM for keep informing agents about their updated contextual norms.

Table 3. The core of the DynaCROM implementation

```
(1)  Model m = ModelFactory.createDefaultModel();
(2)  Resource configuration =  m.createResource();
(3)  configuration.addProperty ( ReasonerVocabulary.PROPruleSet,
(4)                              ontologyDir.concat ("rulesToComposeNorms.rules") );
```

```
(5)  Reasoner reasoner =
(6)                  GenericRuleReasonerFactory.theInstance().create(configuration);
(7)  InfModel inferredModel = ModelFactory.createInfModel(reasoner, this.getOntModel());
```

3 Case Study

The domain of multinational organizations is used for presenting our case study. This domain was chosen because it well illustrates important implicit contextual information found in open MASs. Fig. 4 illustrates our world, created as follows: USA is an environment that belongs to North America; Cuba is an environment that belongs to Central America; Brazil is an environment that belongs to South America. PUCie-Rio and Dellie Brazil are organizations located in Brazil; Dellie Cuba is an organization located in Cuba; Dellie Brazil and Dellie Cuba are branches of the Dellie headquarters, which is located in USA. All Dellie organizations define the supplier and customer roles; PUCie-Rio defines only the customer role. Dellie organizations sell computers; PUCie-Rio is a university.

Environment Organization	Organization Role
North America USA Dellie	Dellie Dellie supplier Dellie customer
Central America Cuba Dellie Cuba	Dellie Cuba Dellie Cuba supplier Dellie Cuba customer
South America Brazil Dellie Brazil PUCie-Rio	Dellie Brazil Dellie Brazil supplier Dellie Brazil customer PUCie-Rio PUCie-Rio customer

Fig. 4. The environments, organizations and roles created for our case study

3.1. Examples of Environment, Organization, Role and Interaction Norms

Usually, organizations do not make their norms public because they are of strategic importance to their businesses. Because of this, we created the following environment, organization, role and interaction norms based on the available information collected from several corporate Web sites.

3.1.1. Examples of Environment Norms:

a. In Central America, if the deliver address is outside one of its environments, every shipped order is obliged to have its price increased by 15% as taxes.

b. In Cuba, all negotiations are obliged to be paid in Cuban pesos (CUP), its national currency. Negotiations outside Cuba are obliged to have their values converted from CUP to the national currency of the country in which the seller is located.

c. In Brazil, all negotiations are obliged to be paid in Reais (R$), its national currency. Negotiations outside Brazil are obliged to have their values converted from R$ to the national currency of the country in which the seller is located.

d. In USA, all negotiations are obliged to be paid in American dollars (USD), its national currency. Negotiations outside USA are obliged to have their values converted from USD to the national currency of the country in which the seller is located.

3.1.2. Example of Organization Norms:

a. Dellie organizations are obliged to ask Dellie headquarters the prices of its products for every large order placed (more than 100 items).

b. Dellie organizations are prohibited to deliver orders during holidays in their final destinations.

3.1.3. Example of Role Norms:

a. In Dellie Brazil, sellers are obliged to ship complete orders on their due dates.

b. In Dellie Cuba, sellers are prohibited to offer more than 8% as discounts.

3.1.4. Example of an Interaction Norm:

a. In Dellie Cuba, customers are obliged to make a down payment of 10% for every order placed to a seller.

3.2. Representing Our Created World

The DynaCROM ontology was extended and instantiated, by using the Protégé Editor [31], for representing the world of our case study. As an example, the *Environment (DynaCROM)* concept was extended with the "*Continent*" and "*Country*" *(domain)* sub-concepts. Thus, "*NorthAmerica*", "*CentralAmerica*", and "*SouthAmerica*" were created as instances of the "*Continent*" concept; and, "*USA*", "*Cuba*" and "*Brazil*" were created as instances of the "*Country*" concept.

For explaining how domain contextual norms are represented, we will use the organization norm 3.1.2.b as motivation. For this norm, precise information about holidays is an important data. Environments can have both *federal holidays*, which are applied to all cities from a country, and *city holidays*, which are only applied for a city. Yet, these holidays can be in the same dates, as Christmas Day (December, 25) and New Year's Day (January, 01), or in different dates, as Independence Day (e.g.,

September, 07 in Brazil and July, 04 in USA) and Labor Day (e.g., May, 05 in Brazil and in the first Monday of September in USA). For representing the information about holidays, the "*Holiday*" concept with its "*FederalHoliday*" and "*CityHoliday*" sub-concepts were created in a DynaCROM domain ontology. Then, these concepts were instantiated for supporting the organization norm 3.1.2.b. For instance, Fig. 5 illustrates the *city* and *federal holidays* created for a city called "*RioDeJaneiro*" located in "*Brazil*".

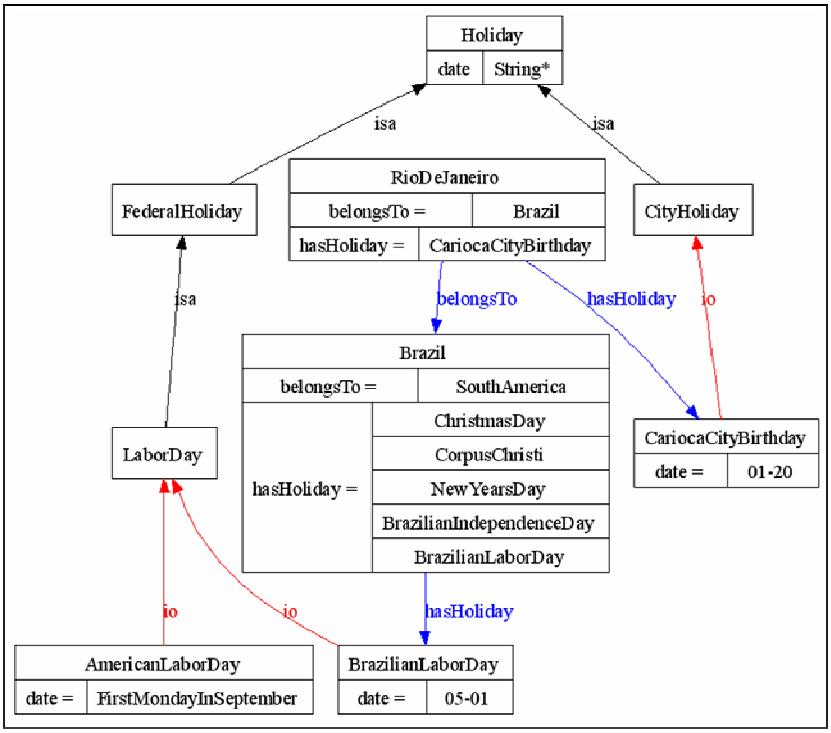

Fig. 5. Part of a DynaCROM domain ontology, extended and instantiated

As previously mentioned, domain rules can be freely created and joined with DynaCROM rules. Table 4 illustrates a domain rule (`Rule6`), which states that a given city will have its holidays composed with the holidays of its country. More precisely, the following process is executed: in (25), the country (`?Country`) of the given city (`?City`) is discovered; in (24), the holidays of the country (`?FederalHolidays`) are discovered; finally, in (23), the holidays from the country are composed with the holidays of the given city. For instance, regarding the organization norm 3.1.2.b and that PUCie-Rio is an organization in a city called "*RioDeJaneiro*" (located in "*Brazil*"), `Rule6` provides the information that Dellie suppliers are prohibited to deliver PUCie-Rio orders during the following holidays: "*CariocaCityBirthday*", "*ChristmasDay*", "*CorpusChristi*", "*NewYearsDay*", "*BrazilianIndependenceDay*" and "*BrazilianLaborDay*" (see these instances in Fig. 5).

Table 4. A rule for composing *city* and *federal holidays*

```
(22) Rule6- [ruleForCityWithFederalHolidays:
(23)          hasHoliday(?City,?FederalHolidays)
(24)            <- hasHoliday(?Country,?FederalHolidays),
(25)               belongsTo(?City,?Country)]
```

3.3. Implementation

Our case study was implemented in JAVA, using JADE and the JENA API [19]. *JADE containers* were used for representing the abstractions of environments and organizations. Agents were implemented extending the *JADE Agent class* with both an attribute for agents' locations and two specific behaviors. One behavior is called *Migratory* and it makes agents move randomly from one location to another. The other behavior is called *Normative* and it continuously informs agents about their current contextual norms, representing the DynaCROM core. Once an agent migrates, its location attribute is updated and, consequently, the answers from its *Normative* behavior change for informing its new contextual norms. Moreover, because DynaCROM is implemented as an active *JADE behavior*, it always executes the process illustrated in Fig. 3. Thus, if any norm is updated in a DynaCROM ontology instance or if any new composition of contextual norms is done in a DynaCROM rule file, agents concerned with these changes will automatically receive different answers.

Fig. 6 illustrates the *JADE containers* created for representing the USA, Cuba and Brazil environments and for representing the Dellie, Dellie Cuba and Dellie Brazil organizations. These containers offer possible locations for mobile agents to go. For instance, an agent called "*****MobileAgent", which has the *Migratory* and *Normative* behaviors, is in Cuba. There, DynaCROM informs the agent about environment norms 3.1.1.a and 3.1.1.b. If the agent migrates to Dellie Brazil, then, DynaCROM informs the agent about environment norms 3.1.1.c and 3.1.1.d, organization norms 3.1.2.a and 3.1.2.b, and role norm 3.1.3.a. All informed norms are in compliance with the norms of our case study, DynaCROM hierarchical form and agent contexts.

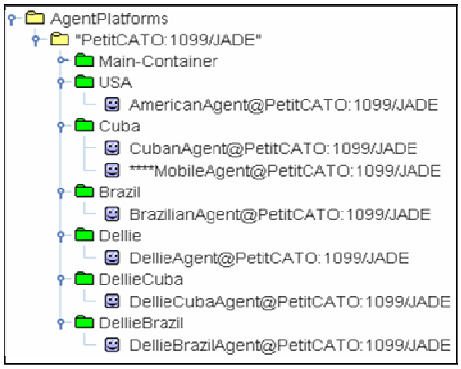

Fig. 6. *JADE containers* for representing our environments and organizations

4 Using Contextual Norm Information

In the current version of DynaCROM, norms are not enforced. DynaCROM keeps informing them to agents, who are free to decide if they will or not use the information. However, DynaCROM can have its output (agents' updated contextual norms) used as a precise input for norm enforcement solutions and, in turn, it can make use of the outputs from these solutions (e.g., information about agents' violated norms).

4.1. Using DynaCROM Output as an Input for a Norm Enforcement Solution

We are currently studying both how DynaCROM output can be used as a precise input for a norm enforcement framework and what DynaCROM can have back from this framework. The chosen framework is called SCAAR (meaning *Self-Controlled Autonomous Agents geneRator*) [6] and it enhances agents with a self-monitoring capability for avoiding norm violation. Because the current version of SCAAR is implemented in SICStus Prolog [29], we still could not use it as a fully norm enforcement mechanism for DynaCROM (implemented in JAVA). However, we are already being able to use SCAAR for informing DynaCROM about norm violations.

Fig. 7 illustrates how DynaCROM and SCAAR are working together. DynaCROM is responsible for continuously informing SCAAR about the norms of agents, according to their current contexts. SCAAR uses this information as a precise updated input instead of using general and pre-defined *(outdated)* information. SCAAR keeps verifying norm compliance and, if a norm is violated, SCAAR informs it to DynaCROM.

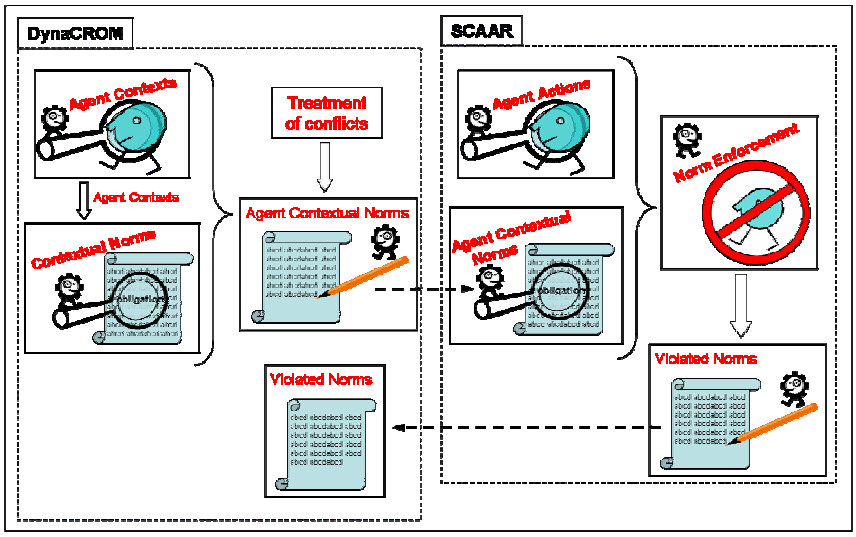

Fig. 7. DynaCROM and SCAAR working together

4.2. Using a Norm Enforcement Solution for Detecting Norm Violation

For exemplifying how DynaCROM and SCAAR can represent a powerful complementary solution while detecting norm violation in open MASs, a simple (not completed) scenario is proposed. The scenario is created according to the world and norms from Section 3 and it can be summarized by the following steps:

a. A Dellie Brazil supplier receives a large order (1500 computers) from PUCie-Rio.

b. Dellie Brazil does not have all the CPUs necessary to build the computers ordered. So, 500 more CPUs will need to be bought.

c. The Dellie Brazil supplier decides to buy the missing 500 CPUs from Dellie Cuba.

d. The Dellie Brazil supplier asks Dellie the price of each CPU.

e. Dellie answers to the Dellie Brazil supplier the price of US$100 for each CPU.

f. The Dellie Brazil supplier multiplies the value (in US$) of each CPU by 500 and converts the value to CUP (Cuban Pesos), the Cuban national currency. The final price for the order is: US$100 * 500 = US$5000 * CUP1 = CUP5000.

g. The Dellie Brazil supplier (being a Dellie Cuba customer) sends the order to a Dellie Cuba supplier with the value of CUP500 as a down payment for the placed order.

h. The Dellie Cuba supplier informs the Dellie Brazil supplier that the value for the down payment is wrong because it is necessary to increase the final price by 15% as taxes. The correct value for the down payment is: (CUP5000 +15%) *0.1 = (CUP5750) *0.1 = CUP575.

After step 4.2.c from the scenario above, all the following steps have contextual norms associated to them. These norms are re-written by DynaCROM according to both SCAAR syntax and current agent contexts and send, in a sequence order one by one, from DynaCROM to SCAAR as different inputs (see Fig. 7).

Table 5 presents the SCAAR contextual norms written for the proposed scenario. `SCAARNorm1` (lines 1 – 4) represents the DynaCROM organization norm 3.1.2.a for regulating step 4.2.d. `SCAARNorm2` (lines 5 – 9) represents the DynaCROM environment norm 3.1.1.d for regulating step 4.2.e. `SCAARNorm3` (lines 10 – 13) represents the DynaCROM environment norm 3.1.1.b for regulating step 4.2.f. `SCAARNorm4` (lines 14 – 18) represents the DynaCROM interaction norm 3.1.4.a for regulating step 4.2.g. `SCAARNorm5` (lines 19 – 24) represents the DynaCROM environment norm 3.1.1.a for regulating step 4.2.h.

Table 5. SCAAR contextual norms

```
(1)  SCAARNorm1-
(2)  [(agt: aDellieSupplier)
(3)    OBLIGED (agt do askPrice with receiver = Dellie)
(4)    BEFORE  (agt do informPrice with quantity > 100)]

(5)  SCAARNorm2-
(6)  [(agt: aSupplier)
```

```
(7)    FORBIDDEN (agt do informPrice with
(8)                   currency # usDollars)
(9)    IF (agt be location with country = USA)]

(10)  SCAARNorm3-
(11) [(agt: aSupplier)
(12)   FORBIDDEN (agt do informPrice with currency # CUP)
(13)   IF (agt be location with country = Cuba)]

(14) SCAARNorm4-
(15) [(agt: aDellieCubaCustomer)
(16)   OBLIGED (agt do giveDownPayment with percent = 10)
(17)   BEFORE  (agt do sendOrder with
(18)             shipperOrganization = DellieCuba)]

(19) SCAARNorm5-
(20) [(agt: aDellieCubaCustomer)
(21)   OBLIGED (agt do addTaxes with percent = 15)
(22)   BEFORE  (agt do sendOrder with shipperCountry # C)
(23)   IF (agt be location with
(24)        situated = Central America and country = C)]
```

SCAAR makes use of DynaCROM inputs (contextual norms) for regulating agent actions. These actions (e.g., "*askPrice*", "*informPrice*", *"giveDownPayment"*, "*sendOrder*", "*addTaxes*" from Table 5) are known *a priori* by SCAAR. Then, SCAAR adds in agent codes both control hooks and an enforcement core. These additions are completely transparent to agents. While an agent is executing, its control hooks *(automatically)* keep informing the enforcement core about the execution of regulated actions. Table 6 presents this algorithm. Then, the enforcement core *(automatically)* keeps verifying if each action is executing according to its norms. If not, it stops the execution of the action and informs the violation to DynaCROM. Verification of norm compliance is done by using Petri nets [25] for representing norms and by following the algorithm presented in Table 7.

Table 6. The SCAAR algorithm for norm enforcement

```
(1)   Let I be information about the agent behaviour.
(2)   Let {t1,...,tn} be the set of transitions associated with I.
(3)   Let {P1,...,Pm} be the set of Petri nets associated with the agent.
(4)   Let {Pact1,...,Pactp} be the set of activated Petri nets (i.e. associated with the
      norms to be manage)
(5)   Let tij be the transition i of the net j.
(6)   for all Pk in {P1,...,Pm} with t1k in {t1,...,tn}}
(7)         Pact(p+1)<- create an instance of Pk
(8)         add Pact(p+1) in {Pact1,...,Pactp}
(9)   end for
(10)  Let {Pact1,...,Pactl} be the set of the activated Petri nets including a tij in
      {t1,...,tn}, j in {1,...,l}
(11)  for all Pactj in {Pact1,...,Pactl}
(12)        inform Pactj of the information associated with tij
(13)  end for
```

Table 7. The SCAAR algorithm for verifying norm compliance

```
(1) Let {t1,..,tn} be the set of transitions of the Petri net.
(2) Let I be the sent information.
(3) Let tI be the transition associated with the information I in {t1,...,tn}
(4) A transition ti is activated if a token stands in all the previous places of ti (in
    SCAAR Petri net, arcs are one-valuated).
(5) if tI is activated then
(6)         if tI is fireable then
(7)             fire the transition tI
(8) else throw exception
```

Returning to our example, SCAAR detects the norm violation that occurred in step 4.2.h by using the algorithms presented in Table 6 and Table 7 as follows: the control hook for the action "*sendOrder*" of the Dellie Cuba supplier sends to the agent enforcement core that the action is being performed. Thus, the agent enforcement core creates instances of Petri nets for representing the norms of the action (`SCAARNorm3`, `SCAARNorm4` and `SCAARNorm5` from Table 5). For instance, P1 (see below) is the Petri net created for representing the norm `SCAARNorm5`.

P1: <P, T, Pre, Post>: ((p_1,p_2,p_3,p_4), ($t_{location}$, $t_{sendOrder}$, $t_{addTaxes}$), (Pre(p_1, $t_{location}$), Pre(p_2, $t_{addTaxes}$), Pre*(p_2, $t_{sendOrder}$)), (Post(p_2, $t_{location}$), Post(p_3, $t_{addTaxes}$), Post(p_4, $t_{sendOrder}$)))

*: it means an inhibitor arc between the transition and the previous place. A transition with an inhibitor arc can be fire when the previous place is empty.

The enforcement core of the Dellie Cuba supplier keeps waiting the information about $I_{location}$, $I_{addTaxes}$ and $I_{sendOder}$ from the associated control hooks of the agent. When the agent arrives in Cuba, the enforcement core receives the information about $I_{location}$. Thus, the Petri net P1 is activated and the transition is fired by putting the Petri net token in the next place (p_2). When the Dellie Brazil supplier tries to perform the action "*sendOrder*", its enforcement core blocks the execution of the action, because the transition $t_{addTaxes}$ was not yet executed (the Petri net token didn't pass through p_2 before being in p_3), and throws an exception.

5 Related Work

García-Camino *et al.* [14] propose a distributed architecture to endow MASs with a social layer, in which normative positions are explicitly represented and managed via rules. Every external agent from the architecture has a *dedicated governor agent* connected to it, enforcing the norms of executed events. DynaCROM also uses rules to manage normative agent positions, however, its focus is on executed actions instead of executed events. Norm enforcement in DynaCROM can be done with few *dedicated governor agents* responsible for monitoring only executed actions. For instance, an open MASs from the traffic domain enhanced with DynaCROM can have only *dedicated governor agents* (e.g., playing the police officer role) for moni-

toring the speed of the cars that pass through regulated crossroads. Thus, it is not necessary to duplicate the number of agents for having the norm enforcement. Moreover, DynaCROM provides a more precise mechanism for norm representation while using contexts.

Vázquez-Salceda *et al.* [34] propose the OMNI framework (*Organizational Model for Normative Institutions*) for modeling agent organizations. Comparing DynaCROM with OMNI, both define a meta-ontology with a taxonomy for norm representation. One difference between the works is that, in OMNI, enforcement is carried out by any internal agent from an MAS; in DynaCROM, enforcement is carried out only by specific trusted agents or by the own regulated agents. A second difference between the works, and the most important, is that, in OMNI, the idea of different levels of abstractions for norms is not explicit, especially for the environment and role levels. On the other hand, DynaCROM is entirely based on different levels of abstractions for norms (its regulatory contexts) for simplifying the tasks of norm management and evolution. For instance, the social structure of an organization in OMNI describes, at the same level of abstraction, norms for roles and groups of roles. Group of roles is used to specify norms that hold for all roles in the group. DynaCROM uses the organization regulatory context to specify organization norms, which hold for all roles from an organization, and it uses the role regulatory context to specify role norms, both regulatory contexts from different levels of abstractions.

6 Conclusion

In this paper, we detail DynaCROM – our ongoing work for providing contextual norm information in open MASs. For agents, DynaCROM keeps informing updated norm information according to their contexts. Norm-aware agents can use the provided norm information for performing correctly and, thus, for achieving their goals faster. For developers, DynaCROM decreases the complexity of norm management in two different cases. The first case is when norms need to be added, updated or deleted. For this case, simply updating the ontology instance concludes the evolution. The second case is when new compositions of contextual norms are desired. For this case, simply activating or deactivating existing rules or creating new ones concludes the evolution. The dynamics for manually customizing several compositions of contextual norms is given by different activations and deactivations of rules, which can be modified at system run-time.

DynaCROM has being used in three different application domains. For the domain of ubiquitous computing [18, 30], DynaCROM has being used in the implementation of context-aware pervasive mobile applications [35]. Instead of using *JADE containers* for simulating environments and organizations, we are using MoCA (Mobile Collaboration Architecture) [27] for delivering updated real location information of mobile devices. MoCA infers mobile devices' locations based on the intensity of their signals to 802.11 network access points. DynaCROM uses MoCA answers (device locations) to continuously apply contextual norms in the agents from the mobile devices. For the domain of next-generation wireless communications [2], DynaCROM has being used for automatically change prices and other parameter values (based on

pre-defined rules) according to overloads in regulated networks. The idea is to keep balancing the use of network bandwidths by distributing clients in particular networks. Clients will be guided to always use a not overloaded network by following pricing discounts. Thus, clients can be better distributed in regulated networks by only changing domain rules and data. For the domain of Brazilian navy [4], DynaCROM has been used for dynamically determining better routes for ships based on climate and other pre-defined conditions.

The current version of DynaCROM has three main points that need improvement. The first improvement is that DynaCROM should deal with conflicts; the second improvement is that DynaCROM should detect norm violations; and, the third improvement is that DynaCROM should enforce norms for avoiding their violation. DynaCROM is not currently addressing the issue (general and difficult) of conflicts, but its modularization of norms helps to make this information more manageable. For the second and third improvements, the SCAAR solution for norm enforcement is being studied. We chose SCAAR instead of LGI (a well-known solution for norm enforcement) [23-24], mainly because SCAAR permits the enforcement of norms that are not related only to agent interactions. Thus, SCAAR will make it possible to enforce DynaCROM environment, organization and role norms independent of the enforcement of interaction norms. For future work, we are planning to implement SCAAR in JAVA in order to fully integrate it with DynaCROM. We believe that DynaCROM and SCAAR can represent together a unique and powerful contextual norm enforcement solution for open MASs.

Acknowledgments

This work was partially funded by the ESSMA (CNPq 552068/2002-0) and EMACA (CAPES/COFECUB 482/05 PP 016/04) projects, and by CNPq individual grants.

References

1. Artikis, A., Pitt, J., Sergot, M.: Animated specifications of computational societies. In Procs. of the AAMAS-2002. Part III. (2002) 1053–1061
2. Berezdivin, R., Breinig, R., Topp, R.: Next-generation wireless communications concepts and technologies. In the IEEE Communications Magazine 40. (2002) 108–16
3. Bouquet, P., Giunchiglia, F., Harmelen, F.v., Serafini, L., Stuckenschmidt, H.: C-OWL: Contextualizing Ontologies. In Procs. of the ISWC. LNCS 2870, Springer. (2003) 164–179
4. Brazilian navy. In <https://www.mar.mil.br/>. (2006)
5. Castelfranchi, C., Dignum, F., Jonker, C.M., Treur, J.: Deliberative Normative Agents: Principles and Architecture. In Procs. of the ATAL-99. (1999)
6. Chopinaud, C., Seghrouchini, A.E.F., Taillibert, P.: Prevention of harmful behaviors within cognitive and autonomous agents. In Procs. of the ECAI'06. (2006) 205–209.
7. Dey, A.: Understanding and using context. In Personal and Ubiquitous Computing. 5(1). (2001) 4–7. ISSN: 1617-4909
8. Felicíssimo, C.H.: Dynamic Contextual Regulations in Open Multi-agent Systems. In Procs. of the ISWC. LNCS 4273, Springer. (2006) 974–975. ISSN: 0302-9743
9. Felicíssimo, C.H., de Lucena, C.J.P., Briot, J.-P., Choren, R.: Regulating Open Multi-Agent Systems with DynaCROM. In Procs. of the SEAS. (2006)

10. Felicíssimo, C.H., de Lucena, C.J.P., Briot, J.-P., Choren, R.: An Approach for Contextual Regulations in Open MAS. In Procs. of the AOIS. (2006)
11. Felicíssimo, C.H., de Lucena, C., Carvalho, G., Paes, R.: Normative Ontologies to Define Regulations over Roles in Open Multi-Agent Systems. In Procs. of the AAAI Fall Symposium TR FS-05-08. (2005). ISBN 978-1-57735-254-9
12. Ferber, J., Gutknecht, O., Michael, F.: From Agents to Organizations: an Organization View of Multi-Agent Systems. In Procs. of the AOSE. (2003)
13. Garcia-Camino, A., Noriega, P., Rodríguez-Aguillar, J.A.: Implementing Norms in Electronic Institutions. In Procs. of the AAMAS. (2005). 2:667–673
14. García-Camino, A., Rodrígurez-Aguilar, J.A., Sierra, C., Vasconcelos, W.: A Distributed Architecture for Norm-Aware Agent Societies. In Procs. of the DALT'05. (2005)
15. Gosling, J., Joy, B., Junior, G.L.S., Bracha, G.: The Java Language Specification. In: <http://java.sun.com/>. (2006). ISBN 0-201-31008-2
16. Grizard, A., Vercouter, L., Stratulat, T., Muller, G.: A peer-to-peer normative system to achieve social order. In Procs. of the COIN@AAMAS. (2006)
17. Gruber, T. R.: A translation approach to portable ontology specifications. In Knowledge Acquisition, 5 (2). (1993). 199–220. ISSN: 1042-8143
18. Henricksen, K., Indulska, J.: Developing context-aware pervasive computing applications: models and approach. Pervasive and Mobile Computing, In Press, Elsevier (2005)
19. Jena. In: <http://jena.sourceforge.net/>. (2006)
20. Jennings, N. R.: On Agent-Based Software Engineering. In AI 117(2). (2000) 277–296
21. Jennings, N., Sycara, K., Wooldridge, M.: A Roadmap of Agent Research and Development. In the Journal of Agents and Multi-Agent Systems. (1998) 1:7-38
22. Khedr, M, Karmouch, A.: ACAI: Agent-Based Context-aware Infrastructure for Spontaneous Applications. In Journal of Network & Computer Applications 28(1). (1995) 19–44
23. Minsky, N.H.: The imposition of protocols over open distributed systems. In IEEE Transactions on Software Engineering. (1991)
24. Minsky, N.H.: LGI. In: <http://www.moses.rutgers.edu/>. (2006)
25. Murata, T.: Petri nets: Properties, analysis and applications. In IEEE 77(4). (1989) 541–580
26. Noy, N; Rector, A. (Eds.). Defining N-ary Relations on the Semantic Web: Use with Individuals. In: <http://www.w3.org/TR/swbp-n-aryRelations/>. (2006)
27. Rubinsztejn, H. K., Endler, M., Sacramento, V., Gonçalves, K., Nascimento, F. N.: Support for context-aware collaboration. In Procs. of the MATA 5(10). (2004). 34–47
28. Silva, V.T.da.: From a conceptual framework for agents and objects to a multi-agent system modeling language. Ph.D. Thesis. Port. 252 p. PUC-Rio. (2004)
29. SICStus Prolog. In: <http://www.sics.se/isl/sicstuswww/site/>. (2006)
30. Soldatos, J., Pandis, I., Stamatis, K., Polymenakos, L., Crowley, J.L.: Agent based middleware infrastructure for autonomous context-aware ubiquitous computing services. In the Journal of Computer Communications. (2006)
31. Stanford University School of Medicine: Protégé. In: <http://protege.stanford.edu/>. (2006)
32. Thomas, G., Williams, A.B.: Roles in the Context of Multiagent Task Relationships. In Procs. of the AAAI Fall Symposium TR FS-05-08. (2005) ISBN 978-1-57735-254-9
33. Tilab Co. JADE - Java Agent DEvelopment Framework. In: <http://jade.tilab.com> (2006)
34. Vázquez-Salceda, J.; Dignum, V.; Dignum, F.: Organizing Multiagent Systems. In the Journal of Autonomous Agents and Multi-Agent Systems 11 (3). (2005) 307–360
35. Viterbo, J., Felicissimo, C., Briot, J.-P., Endler, M., Lucena, C.: Applying Regulation to Ubiquitous Computing Environments. In Procs. of the SEAS. (2006)
36. Weyns, D., Parunak, H.V.D., Michel, F., Holvoet, T., Ferber, J.: Environments for Multiagent Systems State-of-the-Art and Research Challenges. In Procs. of the E4MAS. (2004). LNCS 3374, Springer. (2005). 1–47. ISBN 3-540-24575-8
37. Wright, G.H.v.: Deontic Logic. In Mind, New Series, Vol. 60, No. 237. (1951). 1–15