Organizing Multiagent Systems

Javier Vázquez-Salceda

Virginia Dignum

Frank Dignum

institute of information and computing sciences, utrecht university

technical report UU-CS-2004-015

www.cs.uu.nl

Organizing Multiagent Systems

Javier Vázquez-Salceda, Virginia Dignum, and Frank Dignum

Institute of Information and Computing Sciences
Utrecht University,
P.O.Box 80.089, 3508 TB Stretch, The Netherlands
javier/virginia/dignum@cs.uu.nl

Abstract. Despite all the research done in the last years on the development of methodologies for designing MAS, there is no methodology suitable for the specification and design of MAS in complex domains where both the agent view and the organizational view can be modelled. Current multi-agent approaches either take a centralist, static approach to organizational design or take an emergent view in which agent interactions are not pre determined, thus making it impossible to make any predictions on the behavior of the whole systems. Most of them also lack a model of the norms in the environment that should rule the (emergent) behavior of the agent society as a whole and/or the actions of individuals. In this paper, we propose a framework for modelling agent organizations, OMNI , that allows the balance of global organizational requirements with the autonomy of individual agents. It specifies global goals of the system independently from those of the specific agents that populate the system. Both the norms that regulate interaction between agents, as well as the contextual meaning of those interactions are important aspects when specifying the organizational structure.

1 Introduction

Traditional multi-agent models often assume an individualistic perspective on the environment. Agents are taken as autonomous entities pursuing their own individual goals based on their own beliefs and capabilities. In this perspective, global behavior emerges from individual interactions and therefore the final behavior of the whole system cannot be predicted, easily managed or specified externally. However, in critical applications such as those within business environments or government agencies (hospitals, police, justice, etc.), the behavior of the global system must be taken into account and structural characteristics of the domain have to be incorporated. That is, the design of the agent society must consider organizational characteristics such as stability over time, some level of predictability, and commitment to aims and strategies, etc.

Recent developments recognize that the modelling of interaction in MAS cannot simply rely on the agent's own architectures and (communicative) capabilities. Furthermore, organizational engineering of MAS cannot always assume that participating agents will act according to the needs and expectations of the system design. Concepts as organizational rules [41], norms and institutions [9], [13], and social structures [26] arise from the idea that the effective engineering of MAS needs high-level, agent-independent concepts and abstractions that explicitly define the organization in which agents live [42]. These are the rules and global objectives that govern the activity of an enterprise, group, organization or nation. Such characteristics are often specified top-down and imposed on the participants.

Global characteristics are external to each individual agent and independent from the goals and behavior of the agent itself, and therefore cannot easily be incorporated in a multi-agent architecture that starts from an individualistic perspective. This implies that, to a certain degree, agent societies must be pre-established, and organizational design cannot be completely left to the result of autonomous interaction. On the other hand, the volatility and dynamics of organizational environments stresses the need for models and systems that accommodate changes required by new or different organizational aims with a minimum impact on the already existing services. From the organizational point of view this creates a need to check conformance of the actual behavior of the society to the behavior desired by the organization [11].

Apart from the regulations that the organization may impose to its members, in real, complex scenarios (such as e-commerce, e-government, e-care) the environment of the organization usually defines a number of norms and regulations that individual agents and the organization as a whole must meet. However, in most methodologies, these are seen only as extra requirements in the analysis phase of the system. If regulations change (as they usually do from time to time), it becomes very hard to track all the changes to be done in the implementation, as there is no explicit representation of the norms and regulations, but a chain of design decisions that were guided by the norms' requirements. The alternative is to have an explicit representation of the norms. In the agent systems field, most approaches to *Normative Systems* incorporate norms explicitly either represent them at a very low level (policies and procedures) or at a very high, abstract level, formally specifying norms in, e.g., deontic logic. The low level approaches allow an easy implementation, but the problem arises when the correctness of the procedures and policies should be checked against the original regulations. High level approaches are closer to the way regulations are made, so verification is easier to be done. However, high level approaches usually use one or several computationally hard logics like deontic logic ([20, 38]). Although it is possible to capture the norms in this way and even give them a certain kind of semantics to reason about the consequences of the norms, this kind of formalization does not yet indicate how the norm should be interpreted within a certain organization.

The above considerations result in the following list of serious drawbacks of current approaches[1] for their use for (semi-)open MAS:

- too *agent-centric* or too *organizational-centric*: Some methodologies (such as GAIA [40]) are agent-centric, that is, are mainly focused on the model of single agents, and give limited support to model the dynamic interactions of the agents in the agent society. On the other hand, methodologies such as SODA [24] and ISLANDER [13] are too society-centric, in the sense that they completely fix agent interactions in rigid protocols and interfaces such that the agents have no room for autonomous behavior.
- no distinction between *roles* and *active entities* (agents): this is important in order to establish a difference between organizational values and individual (agent) values.
- *normative aspects* are often not considered, or are either *too theoretical* or *too practical*. Few agent methodologies cover normative aspects, and they usually do it by trying to model the whole normative environment in only *one level of abstraction*, either too theoretical (by means of computationally hard logics) or too practical (policies, protocols).
- *ontologies* are seen as an external (accessory) component, while in fact they are tightly coupled with the rest of the framework and are needed to model most of the elements in it.

In order to remedy all the drawbacks listed above we propose a new framework, the Omni framework. Omni (Organizational Model for Normative Institutions) brings together some aspects from two existing frameworks: OperA and Harmon*IA*. OperA [8] is a formal specification framework that focuses on the organizational dimension, properly modelling not only organizational structures in an agent society (structuring the global behavior of the society) but also the aims and behavior of the agents from the agent perspective. It also explicitly provides for ontological descriptions of agent interactions. It therefore provides a possible solution to the first, second and fourth problem signalled above. Harmon*IA* [37] is a formal framework to model especially highly regulated electronic organizations from the abstract level where norms usually are defined to the final protocols and procedures that implement those norms. It also incorporates ontologies to describe and connect the different levels of norms. So, Harmon*IA* clearly tackles the third and fourth drawback above.
By combining the two approaches in Omni we will show that all of the above problems can be adequately solved. Although combinations of frameworks usually tend to get overly complex and difficult to use for specific applications, we will see that the Omni framework is highly modular and depending on the type of application modules are more or less expansively used.

The Omni framework is composed of three dimensions:

[1] A deeper analysis of other approaches is done in section 8

– the **Normative Dimension** of the organization, which specifies the mechanisms of social order, in terms of common norms and rules, that members are expected to adhere to.
– the **Organizational Dimension** of the organization, which describes the structure of an organization, an can therefore be viewed as a means to manage complex dynamics in societies.
– the **Ontological Dimension**, which defines environment and contextual relations and communication aspects in organizations.

Figure 1 depicts the Omni framework. All three dimensions can be considered at different abstraction levels. This facilitates the analysis and design of organizations, by providing the means to link abstract concepts to its implementation counterparts. These abstraction levels are:

– the **Abstract Level**: where the statutes of the organization to be modelled are defined in a high level of abstraction. This level can be used to model elements from a first step in the requirement analysis. It also contains the definition of terms that are generic for any organization (that is, that are not contextual) and the ontology of the model itself.
– the **Concrete Level**: where specific modelling elements are specified, based on the domain analysis and design processes. The meaning of the abstract values defined in the previous level, is refined in terms of norms and rules, roles, landmarks and concrete ontological concepts.
– the **Implementation Level**: represents the implementation phase of the development process. This level assumes a given multi-agent architecture as basis for the implementation of the organizational model, and includes mechanisms for role enactment and norm enforcement.

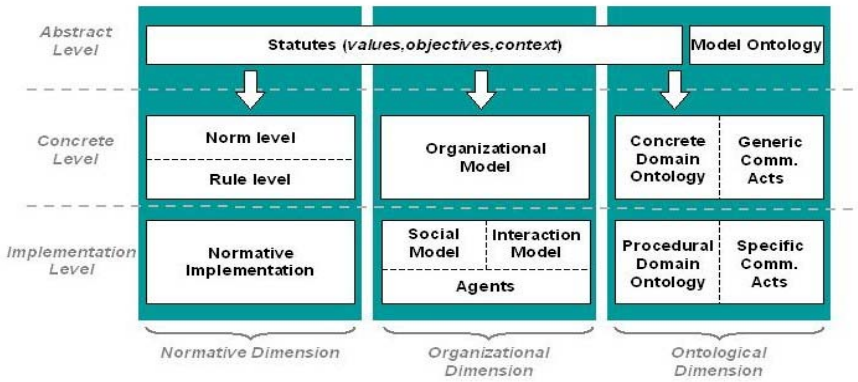

Fig. 1. Levels and dimensions in the Omni framework.

The division of the system into these three levels aims to ease the transition from the very abstract statutes, norms and regulations to the very concrete protocols and procedures implemented in the system. This is especially important in the normative dimension in order to fill the gap between theoretical (abstract) approaches and practical (concrete) ones.

1.1 Organization of this paper

This document is organized as follows. In §2 we will discuss the definition of statutes. Then we will focus on the description of the Organizational dimension (§3) and the Normative dimension (§4) of the organization. Subsequently we will outline in §5 the kind of ontologies and communication languages needed in the Ontological Dimension. In §6 we will study the design of MAS from the agent perspective. In §7 we will describe how to design MAS with Omni, including an example in a highly regulated domain, and in §8 we analyze the relationship between existing approaches and the Omni framework. We end this article with some final conclusions and outline future lines of research.

1.2 Working Examples

In the remainder of this paper, we will illustrate the different components of a society using two examples. The first example, the conference scenario, concerns a domain with a limited normative element. In this scenario, the organization has as main global objective the realization of conferences.

The second example, the organ and tissue allocation scenario, concerns a highly-regulated domain, as the relative scarcity of donors has led to the creation of international coalitions of transplant organizations. In this geographically distributed environment there is the necessity to accommodate a complex set of, in some cases conflicting, national and international regulations, legislation and protocols governing the exchange of organs. These regulations also change over time, making easy-to-adapt solutions of the utmost importance.

2 The Statutes of an Organization

Just as in other software paradigms, a first analysis step is needed in order to analyse the requirements that the final MAS implementation should meet. In our case this first step is done by the definition of the *Statutes* of the organization. Statutes define the foundations of the organization, that is, they are the origin of the definition of the whole organization.

In our framework *statutes* indicate, at the most abstract level, the main *objectives* of the organization, the *values* that direct the fulfilling of this objective and they also point to the *context* where the organization will have to perform its activities.

For example, imagine a conference organized in the context of a research consortium such as the International Foundation for Multi-Agent Systems [2]. The statutes of IFMAS state the following:

> *The International Foundation for Multiagent Systems (IFMAS) is a non-profit corporation whose purpose is to promote science and technology. In pursuit of its purposes, IFMAS will engage in activities including, but not limited to:*
> - *Coordinating and arranging seminars on artificial intelligence and multi-agent systems;*
> - *Becoming a representative forum for experts within the field of artificial intelligence and multi-agent systems;*
> - *Distributing, and making available, knowledge about multi-agent system technology through publications, organizational seminars, courses, and conferences;*
> - *Collaborating with scientific and other institutions, organizations and other societies, including industrial companies, governments, and international bodies with similar or related purposes.*

In this statement we can find:

1. the *objectives*: the main objective of IFMAS is to promote science and technology. Another objective is the organization of seminars.
2. the *context*: IFMAS states that it operates only in the area of artificial intelligence and multi-agent systems.
3. the *values*: The IFMAS is a *non-profit* organization. Implicit in the latter part, it also says that knowledge sharing and distribution are also values of the organization. These values will play a role in the regulations that will determine the actual process according to which seminars should be organized.

The *statutes* of a given organization can be described using a tuple

$$statutes = \langle values, objectives, context \rangle$$

where:

[2] http://www.cse.cs.edu/research/cit/IFMAS/IFMAS.html

- $values = \langle value_1, value_2, \ldots, value_n \rangle$ is the set of *Values* of the organization,
- $objectives = \langle objective_1, objective_2, \ldots, objective_m \rangle$ is the set of objectives the organization aims to fulfill,
- $context = \langle eOrg_1, eOrg_2, \ldots, eOrg_l \rangle$ is the set of organizations that influence the behavior of the organization. The set *context* can be void ($l = 0$).

The *objectives* of the organization express the overall goals of the society. Objectives are expressed by terms (i.e. predicates with parameters). As far as the organization has control over the actions of the agents acting within that organization, it will try to ensure that they perform actions that will lead to the overall goal of the society. We will see in §3.1 and §3.2 how these objectives influence the design process in the *Organizational Dimension*.

The *values* of the organization define which are the aspects of the world about which norms should be defined and also what the ideal situation is. E.g. having "non-profit" as a value, means that revenues is a relevant aspect for normative behavior and the ideal is that the organization does not make a profit. In our framework, values are the basis of the *Normative Dimension*. However, values do not specify *how*, *when* or in *which* conditions individuals should behave appropriately in any given social setup. This is the part played by the norms and rules, as we will see in §4.1 and §4.2.

The environment of an organization can be seen as consisting essentially of other societies or organizations. These societies form the *context*. Organizations and their environment are interdependent, and each influence the other. An important issue is that of communication and common understanding: *can formal models be developed to describe and analyze the relationship between a system and the environment it is inserted in?* Although the context only plays a minor role in this paper we plan to do further research on the applicability of the concept of linking pin developed by Rensis Likert in management theory [18] to the problem of interaction between the society and its environment.

3 The Organizational Dimension

In OMNI, the Organizational Dimension develops the specification of the social structure of the system. Based on the concerns identified in the Abstract Level, the Concrete Level specifies the structure and objectives of a system as envisioned by the organization, and the Implementation Level describes the activity of the system as realized by the individual agents.

The development of agent societies should recognize two complementary perspectives. On the one hand, society design must capture the structure and requirements of the problem owners, and on the other hand, design must realize that the realization of society activity is dependent on individual. That is, agents must be available that are able and interested in enacting society roles. From the point of view of society design, the reasons why an agent wants to enact a role are often not relevant. However, from the agent's perspective, mechanisms must be developed that allow the incorporation of role characteristics into the agent's architecture. As a summary, the organizational dimension of Omni captures the relations between three models depicted in figure 2:

- The *Organizational Model* (OM) specifies the organizational characteristics of an agent society in terms of social structures (roles) and interaction structures (scene scripts).
- In the *Social Model* (SM), the enactment of roles by agents is fixed in social contracts that describe the capabilities and responsibilities of the agent within the society, that is the agreed way the agent will fulfil its role(s).
- In the *Interaction Model* (IM) concrete interaction scenes are dynamically created by role-enacting agents, based on the interaction scripts specified in the OM. Role enacting agents negotiate specific interaction agreements with each other and fix them in interaction contracts.

The *Organizational Dimension* in Omni covers both the organization and the agent perspective in the design of agent societies. By separating organizational and individual concerns, OMNI

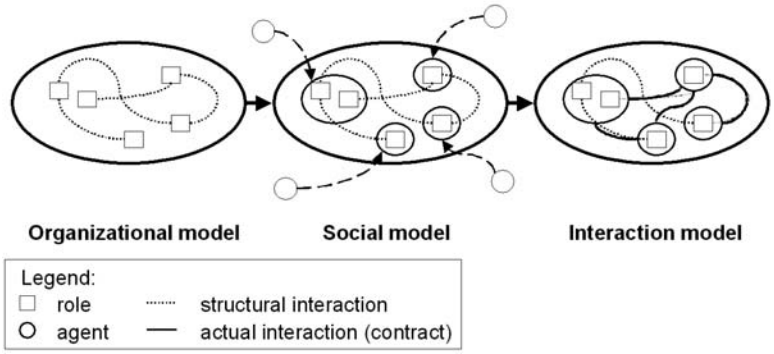

Fig. 2. The three models in the Organizational Dimension.

models are able to respect the autonomy of individual agents while ensuring conformance to organizational aims. *Contracts* are introduced as a means to integrate top-down specification of organizational structures with the autonomy of the participating agents. Contracts must have syntactic, semantic and pragmatic meaning. A language to describe contracts must be rich enough to describe knowledge needs and situations, and also be executable [10].

In the following subsections we describe in more detail the three levels of the Organizational Model.

3.1 The Organizational Abstract Level

The abstract level of the Organizational Dimension describes a social system from the perspective of the organization, that is, which are the aims and concerns of the organization with respect to the social system. At the abstract level, as we saw in §2, this is defined by means of a list of the organization's externally observable *objectives*, that is, the desired states of affairs in the life of the society.

A common way to express the objectives of an organization is in terms of its expected functionality, that is, what is the organization expected to do or produce. The determination of the overall objectives of the society follows a process of elicitation of functional (*what*) and interaction (*how*) requirements. For example, in the conference scenario, the purpose of organizing seminars, expressed in the statutes of IFMAS, is taken as the global objective of the conference society.

In order to identify the objectives of an organization, it is important to characterize the different stake holders of the organization, their requirements, expectations, constraints and relationships to each other. Stake holders are entities or systems in the environment that have goals or expectations towards the society. These need to be identified in order to evaluate their requirements and expectations towards the society. Stake holders form one of the basis for the identification of roles in the concrete level of specification of an organization, as will be discussed in the next subsection.

3.2 The Organizational Concrete Level

The aim of the Organizational Concrete level is to specify the *Organizational Model* (OM) for the society, based on the global objectives and stake holders of the society, expressed in the Abstract Level. The OM describes both the objectives and the means to achieve those objectives, from the perspective of the society owners. The organizational characteristics of an agent society are specified in the OM in terms of two structures:

- *Social structure* (SS): describes the society roles, their relationships, capabilities and activities.
- *Interaction structure* (IS): specifies the abstract processes that according to the organization's view must be used to achieve its objectives. This declarative description specifies the states that agents (enactors of roles) must strive to achieve, instead of the activities to perform.

Any complete society specification must also include the description of concepts and relationships holding in the domain, and of the social values and norms, that is, what is to be accepted as 'good' social behavior. Therefore, as depicted in figure 3, The OM must link to both the role norms, scene norms and transition norms, defined in the concrete level of the Normative Dimension (see §4.2), and the ontologies and communication languages defined in the concrete level of the Ontological Dimension (see §5).

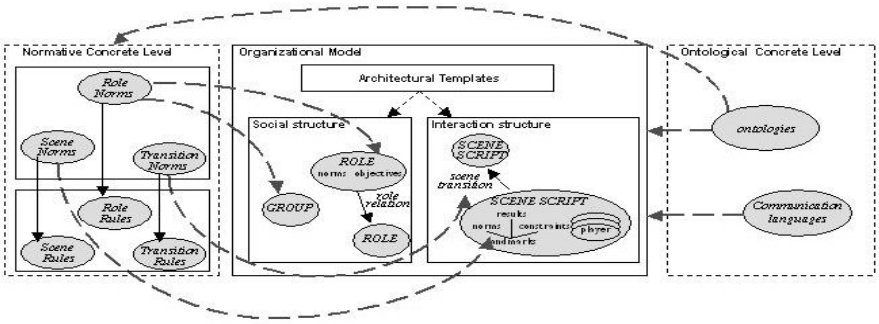

Fig. 3. Interactions between OM and the Normative and Ontological Dimensions.

Finally, the *Architectural Templates* support the definition of the organizational structures by providing three basic coordination patterns (hierarchy, network and market). Different coordination needs require different facilitation approaches, that is, different actors and tasks necessary to facilitate the activity of the organization [12]. In this sense, the choice of a coordination model has great impact in the design of the agent society as it determines how the basic interaction happens. These templates contain a library of role and interaction specifications specific of each basic coordination type, and are used as basis for the development of the OM for a domain.

The Social Structure : The social structure of an organization describes the roles present in the organization, including their objectives, rights and requirements, possible groups of roles, and the relations between roles. Roles identify activities and services necessary to achieve social objectives and enable to abstract from the specific individuals that will eventually perform them. From the society design perspective, roles provide the building blocks for the agent systems that can perform the role, and from the agent design perspective, roles specify the expectations of the society with respect to the agent's activity in the society.

The social structure is composed by the following elements:

- *Roles*: A list of role definitions
- *Role Hierarchy and Role Dependencies*: A list of triples of two role names and the name of the relationship between them
- Groups: list of sets of roles

Roles are the main element of the *Social Structure*. Role descriptions should identify the activities and services necessary to achieve society objectives and enable to abstract from the individuals that will eventually perform the role. In Omni the definition of a role is composed of the following elements:

- *Role id*: A unique name with which to refer to this role
- *Objectives*: A set of landmarks that describe the desired results of this role
- *Sub-objectives*: A set of landmarks that described desired intermediate states for role objectives
- *Rights*: A set of expressions identifying the rights of this role

– *Norms and Rules*: A list of normative expressions that apply to this role. We will see how we formalize them in §4.2.
– *Type*: Either *external* or *institutional*, indicating which type of agents can apply for this role

An example of role description is presented in table 1.

Id	PC_member
Objectives	paper_reviewed(Paper,Report)
Sub-objectives	{read(P), report_written(P, Rep), review_received(Org, P, Rep)}
Rights	access-confmanager-program(*me*)
Norms & Rules	PC_member is OBLIGED to understand English IF paper_assigned THEN PC_member is OBLIGED to review paper BEFORE given deadline IF author of paper_assigned is colleague THEN PC_member is OBLIGED to refuse to review ASAP
Type	external

Table 1. *PC member* role description.

Groups provide means to collectively refer to a set of roles. Moreover, *groups* are used to specify norms that hold for all roles in the group. The elements of a group definition are the following:

– *Group id*: A unique name with which to refer to this group
– *Roles*: A list of role identifiers, specifying the members of the group
– *Norms and Rules*: A list of normative expressions that apply to this role

Members of a *group* must be existing roles in the society. A basic group does not specify any group norms, and is just an efficient way to refer to a group of roles, for example, when a certain interaction scene (defined in the *Interaction Structure*) requires an enactor of one of the roles in the group to be present, without really having to specify which role. A trivial group is *all*, which refers to all roles in the society. However, when norms are specified for a group, these must be consistent with the norms of the roles in the group.

Group id	Organizers
Roles	{PC-Chair, website manager, general chair, local organizer}
Norms & Rules	IF author is member of Organizers THEN author is FORBIDDEN to submit a paper

Table 2. Group description of the *organizer* group.

The distribution of objectives by the roles is represented by means of the definition of a *Role Hierarchy*. Society objectives from the statutes of the organization are the basis for the definition of the objectives of roles, in the sense that the society objectives will be realized by the realization of role objectives. In the same way, as we will see in §4.1 and §4.2, society values determine which norms hold for different roles or groups of roles. Three criteria that can be chosen to guide the determination of the *Role Hierarchy*:

1. One possible option is to divide objectives into those that are related to the coordination of the organization and those that concern the realization of its global objectives. This division defines two types of roles:
   – The *institutional roles*, that is, roles to be enacted by agents that represent and operate the organization, assuring the coordination needs of the organization. These correspond in most of cases with the *facilitation roles* defined in the Architectural Templates.

– The *external roles* can be enacted by agents representing external actors that enter into the organization and are concerned with the realization of the environment-oriented objectives of the society.

Agents performing institutional roles (e.g. *gatekeeper*, or *matchmaker* in an network) are usually developed by the society designers, such that their behavior can be certain to comply with the requirements for the role.

2. Another option is to refine a role by identifying sub-types of this role. This refinement defines a *is-a* relation between sub-roles and roles. For instance, if in the conference scenario we have identified the *participant* role, we can refine it by sub-typing it into *presenters* and *attendants*.
3. We can also refine a role by decomposing it in sub-roles that, together, fulfill the objectives of the given role. This refinement usually defines a *part-of* relation between sub-roles and roles. For instance, if we have identified *conference-organizer* as a role with one or several objectives to fulfill (e.g. organize conference), such an objective can be subdivided in sub-objectives to be fulfilled by different staff members, e.g. *webmaster*, *workshop-chair* and *program-chair*.

Stake holders in this society, are organizer, authors, PC members and participants. The objective of the organizer is to organize a successful conference, authors want to get their papers accepted, the PC member aims at assuring the quality of the program, and the participant hopes for a high quality conference. Facilitation activities can be described in terms of an organizer role that administrates the conference, a chairperson role responsible to regulate conference sessions, etc. The example will be further detailed as the components of Omni are presented. 4 shows the role hierarchy for the Conference scenario.

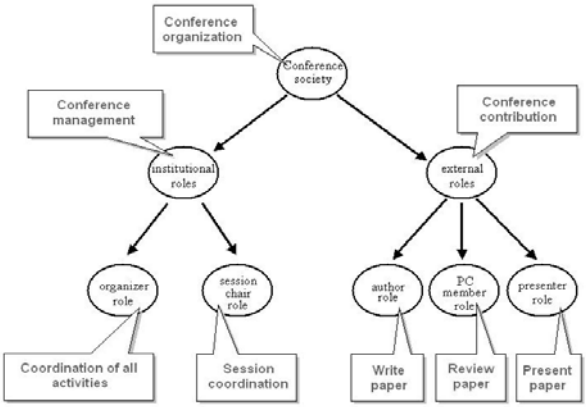

Fig. 4. An example of role hierarchy and objective distribution for the conference scenario.

The refinement process in which sub-roles are determined stops at a level where it is still possible to determine an objective that can serve as a goal for an agent. Mainly this means that the objective should be abstract enough to leave an agent several ways of realizing the objective. When the refining is taken too far the objective gets too specific and the agents that perform the role can only fulfill the objective by performing a certain task or procedure. In that case the agent becomes a simple "traditional" program without any freedom to react to changing circumstances.

Roles are often dependent on other roles for the realization of (part of) their objectives. These relationships are called in Omni *Role Dependencies*. Societies establish dependencies and power relations between roles, indicating relationships between roles. These relationships describe how actors can interact and contribute to the realization of the objectives of each other. That is, an objective of a role can be delegated to, or requested from, other roles. Role dependency between two roles means that one role is dependent on another role for the realization of its objectives. How role objectives are actually passed between two roles depends on the type of coordination

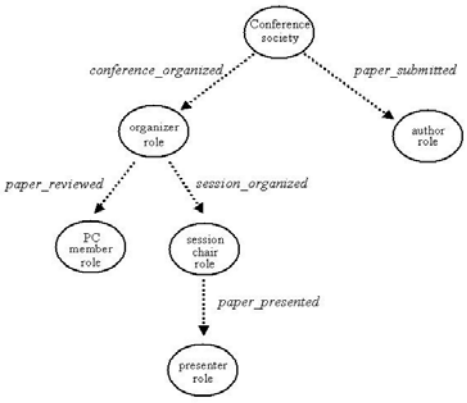

Fig. 5. An example of role dependencies in the conference scenario.

A dependency graph represents the dependency relations between roles. Nodes in a dependency graph are roles in the agent society. Arcs are labelled with the objectives of the parent role for whose realization the parent role depends on the child role. There can be more than one arc between two nodes, representing the fact that the parent role depends on the child role for more than one of its objectives. The root of the graph is the society itself, represented as a super-role. Part of the dependency graph for the conference society is displayed in figure 5. For example, the arc between nodes PC-Chair and PC-member represents the dependency $PC-Chair \succeq_{paper-reviewed} PC-member$. Dependencies between two roles can be established in different ways, and therefore a model must describe how this interaction occurs. The way the objective g in a dependency relation $r_1 \succeq r_2$ is actually passed between $r1$ and $r2$ depends on the coordination type of the society, defined in the Architectural Templates. In OMNI , these templates define three types of role dependencies -*bidding*, *request* and *delegation*- related to the *hierarchy*, *market* and *network* templates, respectively (see table 4).

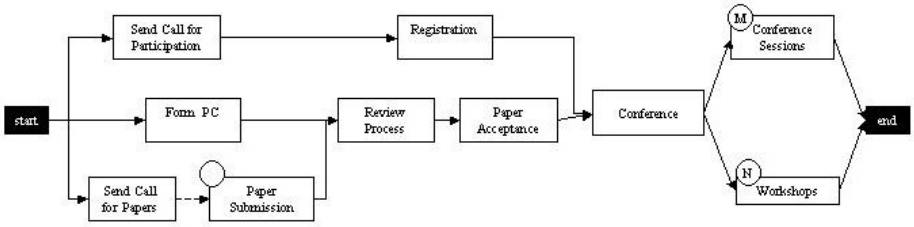

Fig. 6. Interaction Structure in the Conference scenario.

The Interaction Structure In our framework interaction is structured in a set of meaningful scenes that follow pre-defined abstract scene scripts. Examples of scenes are the registration of participants in a conference, which involves a representative of the organization and a potential participant, or paper review, involving program committee members and the PC chair. The relation between scenes is represented by the *Interaction Structure* diagram (see figure 6). In this diagram, *transitions* describe a partial ordering of the scenes, plus eventual synchronization constraints. Note that several scenes can be happening at the same time and one agent can participate in different scenes simultaneously. Transition scripts must furthermore also describe the conditions

for the creation of a new instance of the scene. For each scene, the interaction structure also specifies an upper bound for the number of instances of that scene that are allowed simultaneously. Furthermore, the enactment of a role in a scene may have consequences for the further enactment of roles in following scenes. In these cases the *evolution relations* between roles must be described. Evolution relations specify the constraints that hold for the role-enacting agents as they move from scene to scene in the animated society (see figure 7).

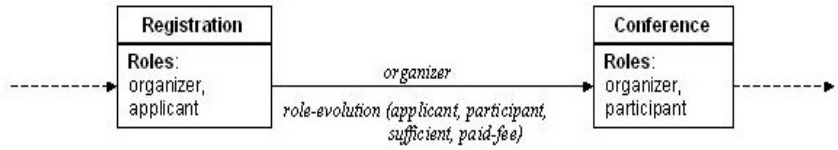

Fig. 7. Role evolution relation for the *applicant* role.

A *scene script* describes an scene by its players (roles), its desired results and the norms regulating the interaction. In the OM, scene scripts are specified according to the requirements of the society. The results of an interaction scene are achieved by the joint activity of the participating roles, through the realization of (sub-)objectives of those roles. A scene script establishes also the desired *interaction patterns* between roles, that is, a desired combination of the (sub-) objectives of the roles.

Scene	Review Process
Roles	Program-Chair (1), PC-member (2..Max)
Results	$r_1 = \forall P \in Papers : reviews\_done(P, review1, review2)$
Interaction Patterns	$PATTERN(r_1)$= {DONE(O,$paper\_assigned(P, PC1, DeadlineR)$ BEFORE DeadlineA), DONE(O,$paper\_assigned(P, PC1, DeadlineR)$ BEFORE DeadlineA), DeadlineA BEFORE DeadlineR, DONE(PC1, $review\_paper(P, Rev1)$ BEFORE DeadlineR, DONE(PC2, $review\_paper(P, Rev2)$ BEFORE DeadlineR}
Norms & Rules	Program-Chair is PERMITTED to assign papers PC-member is PERMITTED to review papers assigned before deadline

Table 3. Script for the *Review Process* scene.

The expressions describing objectives and sub-objectives of a role can be more or less restrictive on the actor performance. That is, the more aspects that are fixed in the expressions, the less freedom an agent enacting the role has to decide on how to achieve the role objectives and interpret its norms. Following the ideas of [33], we call such expressions landmarks.

Landmarks are conjunctions of logical expressions that are true in a state. Several different specific actions can bring about the same state, and therefore, landmarks actually represent families of protocols. The use of landmarks to describe activity enables the actors to choose the best applicable actions, according to their goals and capabilities. The level of specification of landmarks determines the degree of freedom the actors have about their performance.

Therefore, a scene script can be graphically represented by its landmarks and the precedence relations specified in the interaction patterns. Figure 8 is an example of such representation for the case of the Review Process scene.

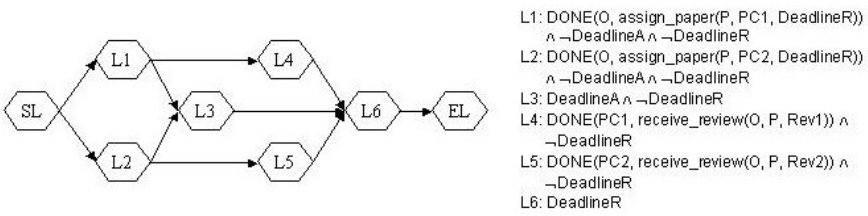

Fig. 8. Landmarks for the *Review Process* scene in the Conference scenario.

The Architectural Templates Different application contexts exhibit different needs with respect to coordination, and the choice of a coordination model will have great impact in the design of the agent society. OMNI provides three architectural templates that model different coordination needs. These templates are used as a basis for the specification of the facilitation layer of an organization, which enables the structured coordination of the different roles and the maintenance of the organization itself.

Williamson [39] argues that the transaction costs are determinant for the type of coordination of an organization. Transaction costs will rise when the unpredictability and uncertainty of events increases, and/or when transactions require very specific investments, and/or when the risk of opportunistic behavior of partners is high. When transaction costs are high, societies tend to choose a hierarchical model in order to control the transaction process. If transaction costs are low, that is, are straightforward, non- repetitive and require no transaction-specific investments, then the market is the optimal choice. Powell [27] introduces networks as another possible coordination model. Networks stress the interdependence between different organizational actors and pay a lot of attention to the development and maintenance of (communicative) relationships, and the definition of rules and norms of conduct within the network. At the same time, actors are independent, have their own interests, and can be allied to different networks. Table 4 gives an overview of the characteristics of the different coordination types.

Coordination in *markets* is achieved mainly through a price mechanism in which independent actors are searching for the best bargain. Agents are self-interested, and driven by their own personal goals. Interaction in markets occurs through communication and negotiation.

Hierarchies are mainly coordinated by supervision, that is, actors that are involved in power-dependent relationships act according to routines. Agents are usually fully cooperative, and coordination is achieved through well-defined command and control lines.

Networks achieve coordination by mutual interest and interdependency. Agents, even if self-interested, agree to collaborate in order to achieve a mutual goal, that benefits all.

	Market	**Network**	**Hierarchy**
Type of society	Open	Trust	Closed
Agent 'values'	Self interest	Mutual interest/ Collaboration	Dependency
Coordination	Price mechanism	Collaboration	Supervision
Relation form	Competition	Mutual Interest	Authority
Dependency relation	Bidding	Request	Delegation
Facilitation roles	Matchmaker, Reputation facilitator, Market master	Matchmaker, Gatekeeper, Notary, Monitor	Controller, Interface facilitator
Conflict Resolution	Haggling (Resort to courts)	Reciprocity (Reputation)	Supervision

Table 4. The three architectural templates.

The choice of a structure should be based on their appropriateness for a specific environment. Once the designer identifies the kind of problem to solve, these templates already define some of the facilitation tasks the social structure should provide, with their related definition of roles:

- **Market structures** are well-suited for environments where the main purpose is the *exchange* of some goods. In this case the social framework proposed identifies three tasks to be performed by facilitator agents: *Matchmaking* facilities to keep track of the agents in the system, their needs and mediate in the matching of demand and supply of services; *Identification* and *Reputation* facilities to build confidence for customers and offer a certain degree of guarantees to all its members despite the openness of the system.
- **Network structures** are well-suited for environments where (dynamic) *collaboration* among parties is needed. In this case one of the main facilitation tasks is the one of the *Gatekeeper*, which is responsible for accepting and introducing new agents into the society; *Notaries* are facilitator agents which keep track of collaboration contracts settled between agents, while *Monitoring agents* can check and enforce the rules of interaction that should guide the behavior in the society.
- **Hierarchical structures** are well-suited for environments where the society's purpose is the efficient *production* of some kind of results or goods or the control of an external production system. In these environments a reliable control of resources and information flow requires central entities that manage local resources and data but also needs quick access to global ones. In this scenario two main facilitation tasks are identified: *Controllers*, which monitor and orient the overall performance of the system or a part of it; *Interface agents* responsible for the communication between the system and the *outside world*.

3.3 the Organizational Implementation Level

While in the previous levels of the Organizational Dimension a system is defined from the point of view of the organization, its objectives and its roles, in the Organizational Implementation Level the central component are the agents themselves, their commitments towards the organization (in terms of agreements concerning the enactment of roles) and towards other agents. This level is composed by three parts:

- the *Social Model* (SM), which specifies the role enacting agreements linking the role specifications with the populations of agents.
- the *Interaction Model* (IM), which describes the possible interaction between the agents in the agent population.
- the *Agents* forming the active agent population. We will talk more about them in §6.

Social Model We assume that individual agents are designed independently from the society to model the goals and capabilities of an external entity. In order to realize their own goals, individual agents will join the society as enactors of role(s) described in the organizational model. This means that several populations are possible for each organizational model. Agent populations of the organizational model are described in the social model (SM) in terms of commitments regulating the enactment of roles by individual agents.

In the OMNI framework, agents are seen as autonomous communicative entities that will perform the society role(s) according to their own internal aims and architecture. Because the society designer does not control agent design and behavior, the actual behavior of the society instance might differ from the intended behavior. The only means the society designer has for enforcing the intended behavior is by norms, rules and sanctions. When an agent applies, and is accepted, for a role, it commits itself to the realization of the role objectives and it will function within the society according to the constraints applicable to its role(s). These commitments are specified as *social contracts* that can be used to verify and predict society behavior. Social contracts can be compared to labor contracts between employees and companies. The society can sanction undesirable (wrong) behavior as a means to control how an agent will do its 'job'.

The Social Model specifies the *role enacting agents* (*reas*) that compose the society at a given moment. For each agent, the *rea* describes the agreements relative to its role enactment and reflects the agent's own requirements and conditions concerning its participation in the society. Depending on the complexity of the implemented agents, the negotiation of contract agreements can be more or less free. Nevertheless, making agreements explicit and formal, allows the verification of whether the animated society behaves according to the design specified in the OM. The SM specifies a population of agents in a society, which can be seen as an instantiation of the OM. When all roles specified in the OM are instantiated to agents in the SM, we say that the SM provides a full instantiation of the society; otherwise, it is a partial instantiation.

Informally, social contracts must specify the activity of agents as enactors of society roles, and include aspects such as the specification of the role(s), the time period the contract holds (either in absolute terms: from date to date, or in relative terms: until certain states hold), specific agreements and conditions governing the role enactment, and the sanctions to take when norms are violated (especially if specific sanctions are agreed upon). Given an agent society S, a social contract for agent s enacting role r is defined as a tuple

$$\textit{social-contract} = \langle a, r, CC \rangle$$

where:

- a is an agent,
- $r \in \textit{roles}(S)$ is a role, and
- CC is a set of contract clauses

A special kind of social contract is the *trivial social contract*, which does not specify any clauses. Such social contract indicates that role enactment follows exactly the description of the role in the OM, that is, the agent does not require any deviations to the expected behavior of the role. Furthermore, each agent can have simultaneously more that one social contract with the society, describing all the roles it enacts. For an example of a social contract, we refer once again to the Conference Society. Imagine that agent Anne will take the role of PC-member. An example of a trivial contract for this enactment is represented in table 5, where agent Anne assume the role of PC-member, following the description of the role exactly. Note that the social contract also sets an alias (Anne) to refer to an agent (identified by a FIPA IOR).

Agent alias	Anne
Agent IOR	00000000000001149444C3A464950412F4D54533A312...
Role	*PC-member*
Contract Clauses	{}

Table 5. Social contract for agent *Anne*.

An example of a more complex social contract is shown in table 6, which describes the case that agent Bob will enact the role of PC member negotiated that he will review only three papers, instead of the supposed 5 per PC member.

Agent alias	Bob
Agent IOR	00000000010100000000E3134372E38332E35392E313...
Role	*PC-member*
Contract Clauses	{OBLIGED(Bob, *max-papers-to-review*(3))}

Table 6. Social contract for agent *Bob*.

Interaction Model The Interaction Model accounts for the actual (emergent) behavior of the society at a given moment. Interaction agreements between agents are described in interaction contracts. Usually interaction contracts will 'follow' the intended interaction possibilities specified in the organizational model. However, because of the autonomous behavior of agents, the interaction model must be able to accommodate other interaction contracts describing new, emergent, interaction paths, to the extent allowed by the organizational and social models.

OMNI provides two levels of specification for interactions. The OM provides a script for interaction scenes according to the organizational aims and requirements and the IM, realized in the form of contracts, provides the interaction scenes such as agreed upon by the agents. It is the responsibility of the agents to ensure that their actual behavior is in accordance with the contracts (e.g. using a monitoring agent or notary services provided by the society for that). However, it is the responsibility of the society, possibly represented by some of its institutional roles, to check that the agents fulfill these responsibilities.

The architecture of IM consists of a set of instances of scene scripts (called scenes), described by the interaction contracts between the role enacting agents for the roles in the scene script. An interaction scene results from the instantiation of a scene script, described in the OM, to the reas actually enacting it and might include specializations or restrictions of the script to the requirements of the reas.

An interaction contract describes the conditions and rules applying to interaction between agents in the agent society. That is, the clauses in an interaction contract specify actual instantiations of interaction scene scripts and must indicate the actors involved and the specific agreements and sanctions concerning the scene to be played. The contract must furthermore involve sufficient reas to cover all the needed roles in the scene. Besides the refinement of the script to the desires and characteristics of the agents participating in the scene instance, interaction contracts must describe the protocol agreed by those agents to fulfil the script landmarks. Interaction protocols are the concrete representation of the refinement of scene script landmarks with the particularities imposed by the participants to the specific communicative capabilities of those participants.

Given a society S and a scene $s \in scenes(S)$, an *interaction contract* is defined as a tuple

$$interaction\text{-}contract = \langle a, s, CC, P \rangle$$

where:

- the set of agents $A = \{a \in Agents : rea(a, r, s) | r \in roles(s)\}$,
- CC is a set of contract clauses, and
- P is the protocol to be followed

The set A in this definition represents the set of all agents enacting reas participating in interaction scene s, and CC is a set of deontic expressions describing refinements to the script, that is, possible conditions and deadlines concerning the results and interaction patterns of scene s. Contract clauses are formally represented by LCR expressions. P is the protocol to be followed by the reas. Protocols describe the actual interaction between reas. A rea interaction protocol describes a communication pattern for reas that is conform to the scene script and consists of CA characteristic of the reas. In principle, any protocol language can be used (e.g., DAML+OIL).

In the Conference Society, tables 7 and 8 are examples of interaction contracts for the *Review Process* scene script , where P1 and P2 are the identifiers of the protocols used The first example (table 7) describes a trivial instantiation of the Review Process script with 5 enactors of the PC-member role. In the second example (table 8), the Review Process script is instantiated to 3 enactors of the role PC-member and is made more specific by indicating that in this case, the program chair is obliged to accept all papers for which the reviewers have not given a review by the deadline.

4 The Normative Dimension

The process of building the normative specification of a MAS goes from the statutes of the organization to the norms to the final implementation of those norms in rules and procedures. The

Agent aliases and roles	{pc1 (*PC-Chair*), pc2 (*PC-Chair*), pc3 (*PC-Chair*), Anne (*PC-Chair*), Bob (*PC-Chair*)}
Scene	*review-process*
Contract clauses	{}
Protocol	P1

Table 7. Example of a trivial interaction contract.

Agent aliases and roles	{pc3 (*PC-Chair*), Anne (*PC-Chair*), pc6 (*PC-Chair*),
Scene	*review-process*
Contract clauses	{ IF NOT *reviews-done*(P, Reva, Revb) BEFORE DeadlineR THEN PERMITTED(PC-Chair, *paper-accepted*(P)}, P2) }
Protocol	P2

Table 8. Example of an interaction contract.

translation steps from one level to the following must be described in a formal way, as we aim to be able to verify if a given organization complies to all the norms that are specified in the regulations. The Normative Dimension describes the different levels of abstraction composing the normative framework that regulates the behavior of the agent society. This normative framework is composed by the following levels, depicted in figure 9:

- The *Normative Abstract Level*, where the values, objectives and context of the organization are defined (by its statutes) and then translated into *abstract norms*.
- The *Normative Concrete Level*, composed by two sub-levels:
  - *Norm level*, where abstract norms are refined into more concrete norms that fix the interpretation of (some) predicates and terms in the context of the organization.
  - *Rule level*, where concrete norms are translated into rules to be computed by agents. This translation is needed as norms have no operational semantics (they only define what *ought to be*, but not *how to be done*).
- a *Normative Implementation Level*, where the final mechanisms to follow the rules are implemented. Possible implementation mechanisms in OMNI are *rule interpreters* and *rule translations*. Rule interpreters provide the ability to reason about rules, while translations generate specific procedures that implement the rules. With this dual approach, agents entering into the organization can either follow the protocols, the rules, or both. Doing so we reduce our assumptions on the internal architecture of the incoming agents. [3]

From a *top-down* direction, the connection between normative levels guides the design process. In this sense, the connection from norms to rules to the final procedures guides the design process of the organization, as it identifies the minimum set of restrictions that are defined by the normative framework, and eases the task of checking if the implemented procedures follow such restrictions.

However, the *bottom up* direction is at least as important. Through the explicit links between procedures, rules, concrete and abstract norms, agents can trace the origin of a given protocol and reason in terms of the rules and norms the protocol implements. This allows the definition of *Flexible Normative Agents*, which are able to handle those unexpected situations that a given protocol has not considered, by reasoning in terms of the related rules and adapting their behavior appropriately.

An additional element in the normative dimension are *policies*. Policies are vertical solutions related to one or more *values* present in the abstract level. Policy definition comprises different

[3] This is mandatory in order to apply our framework to open environments with *heterogeneous incoming agents*.

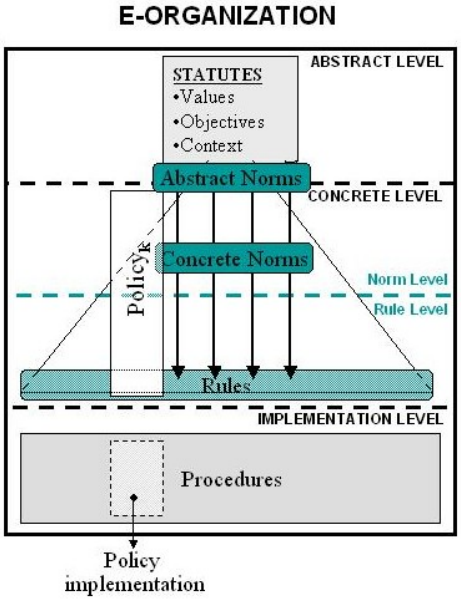

Fig. 9. The abstraction levels in the Normative Dimension.

levels of abstraction, from the abstract norms in the abstract level, to their refinement in *concrete norms* and *rules*. The resulting specification of the policy is then translated into the procedures (e.g., triggers, protocols, etc.) that compose the *implementation of the policy*.

4.1 The Normative Abstract level

At this highest level of abstraction, the *values* fulfill the role of norms in the sense that they determine the concepts that are used to determine the value or utility of situations. However, they do not tell us explicitly how we should behave appropriately in any given social situation. This is the part played by abstract norms, concrete norms and rules (see section 4.2). The values of the Conference Society can be described as 'desires' of the organization. For example:

- the *information sharing* value can be described as $D(share(information))$,
- the *non-profit* value can be described as $D(non\text{-}profit(organization))$,

However, besides a formal syntax, this does not provide any meaning to the concept of *value*. In this paper, we do not intend to define a logic about values, [4], but only use them as an initial step to derive the normative system.

In our framework, the meaning of values is defined by the norms that contribute to this value. In an intuitive way we can see this translation process as follows:

$$\vdash_{org} D(\varphi) \mapsto O_{org}(\varphi)$$

meaning that, if an organization *org* values situations where φ holds more than situations where φ does not hold, then such value can be translated in terms of a norm (an obligation of the organization *org*) to fulfill φ. In our framework a norm *contributes to a value* if fulfilling the norm always leads to states in which the value is more fully accomplished than the states where the norm is not fulfilled. So, each value has attached to it a list of one or several norms that contribute

[4] A more in-depth discussion about the aspects related to values can be found in [21].

to that value. The total list of abstract norms *defines* the meaning of the value in the context of the organization.

$$\vdash_{IFMAS} D(share(information)) \mapsto \{O_{IFMAS}(disseminate(research))\}$$

We define $ANorms$ (the language for abstract norms) to be a deontic logic that is temporal, relativized and conditional, i.e., an obligation to perform an action or reach a state can be conditional on some state of affairs to hold, it is also meant for a certain type (or role) of agents and should be fulfilled before a certain point in time.

For instance, the following norm might hold: *"The authors should submit their contributions before the deadline"*, which can be formalized as:

$$O_{author}(submit(paper) < Deadline)$$

The obligation is directed towards the author, assuming that she is responsible for fulfilling it.

4.2 The Normative Concrete Level

In order to check norms and act on possible violations of the norms by the agents within an organization, the abstract norms have to be translated into actions and concepts that can be handled within such an organization. To do so, the definition of the abstract norms are iteratively concretized into more concrete norms, and then translated into the rules, violations and sanctions that implement them.

The Norm Level The norms at this level are described in $CNorms$ (the language for concrete norms), which we assume for the moment to be equal to $ANorms$, but which might use different predicates. In addition we define a function `I`: $ANorms \rightarrow CNorms$ which is a mapping from the abstract norms to the concrete ones. For each abstract norm `I` indicates how it can be fulfilled by fulfilling concrete norms within the context of this organization. This function is based on the contextual translation operator as developed in [14].

There are several ways in which norms can be *abstract* and thus several ways to make them more concrete. In the following, we present possible concretization issues, using the the contextual translation operator $\rightsquigarrow$. See [14] for more about the contextual translation operator $\rightsquigarrow$.

- **Abstract actions**: Norms often refer to an abstract action that can be implemented in many ways. For example: *"submitting a paper"*. The translation in this case is a kind of definition of the abstract action in terms of the concrete actions. In the above case one could define this as follows:

  $$send\_mail(organizer,\ paper\_files) \cup send\_post(organizer, paper\_hard\_copies)$$
  $$\rightsquigarrow_{IFMAS} submit(paper)$$

  i.e., *"to submit a paper"* is to perform either one of the two more specific actions (to send it electronically or by post). Important to note is that this definition closes the way papers can be submitted.
- **Vague terms**: Norms use terms that are vague (have no precise meaning) and that have to be defined separately. E.g. *"the IFMAS should disseminate the work of scientists on the target area"*. The term *"disseminate"* is so vague that there is a need to refine it in terms that are easier to check and/or implement:

  $$O_{IFMAS}(organize\_seminar(people,\ papers))$$
  $$\rightsquigarrow_{IFMAS} O_{IFMAS}(disseminate(research))$$

  $$O_{IFMAS}(submit\_cfp(people)) \wedge O_{IFMAS}(papers\_reviewed(papers))$$
  $$\wedge O_{IFMAS}(session\_organized(people, papers))$$
  $$\rightsquigarrow_{IFMAS} O_{IFMAS}(organize\_seminar(people, papers))$$

  The dissemination value of IFMAS is made more concrete, as organizing a seminar.

– **Agent and role abstraction**: Norms abstract from the role or agent for whom the norm holds. The process to refine responsibilities is done as roles are refined within the Social Structure's *role hierarchy* (see §3.2):

$$O_{PC\_member}(papers\_reviewed(paper)) \rightsquigarrow_{IFMAS} O_{IFMAS}(papers\_reviewed(paper))$$

– **Temporal abstractness**: Often there is an implicit deadline for obligations, which is implied by the fact that the fulfillment of an obligation is also the fulfillment of a condition for a permission. Returning to the *review_paper* example, there is a missing temporal relation: the obligation of reviewing the paper occurs only if a paper has been assigned, and if so the review should be done before the deadline. We can then extend the formula as follows:

$$done(assign\_paper(P,me,Deadline)) \rightarrow$$
$$O_{PC\_member}(review\_paper(P,\ Rep) < do(pass(Deadline)))$$

– **Actions or situations not directly checkable**: For instance, *"the acceptance or refusal of a paper should only be done taking into consideration the quality of the work, and not any other personal considerations"*. Although the norm is clear, it is impossible to check directly on which basis a decision is taken by an agent. This is an internal (mental) action. Therefore the organization has to devise some constraints and/or procedures that are checkable (or controllable) by the organization and which take care of the fulfillment of the norm. For example, the organization might force reviewers to explicitly give objective reasons for acceptance or refusal.

The Rule Level The translation from norms to rules in OMNI marks a transition from a normative perspective to a more descriptive one. As mentioned before, norms have no operational semantics. Therefore, a descriptive perspective implies a change in the language, from a deontic logic to a language more suitable to express actions and temporal constraints.

In [19], Meyer proposed a reduction from deontic logic to a Propositional Dynamic Logic. In this approach, deontic formulæ such as $O(\alpha)$, $F(\alpha)$ and $P(\alpha)$ are reduced to dynamic logic as follows:

$$O(\alpha) \equiv [\neg\alpha]\, V$$

Informally, it expresses that α is obligatory iff not doing α leads to a violation. An example of this kind of translation in the conference organization is the following:

$$O_{session\_chair}(organize\_session(people,papers)) \equiv [\neg organize\_session(people,\ papers)]\, V$$

Following this idea, each norm can be translated to:

– a violation expression: by using the following reduction rules by Meyer [19]:

$$F(\alpha) \mapsto [\alpha]\, V$$
$$P(\alpha) \equiv \neg F(\alpha) \mapsto \neg\, [\alpha]\, V$$
$$O(\alpha) \equiv F(\neg\alpha) \mapsto [\neg\alpha]\, V$$

– a precedence expression: in those cases where the norm expresses temporal relations among actions, such relation can be also expressed through the [ ] operator as follows:

$$O(\alpha < do(\beta)) \mapsto [\beta]\, (done(\alpha) \vee (\neg done(\alpha) \wedge V))$$
$$O(\alpha < do(\beta)) \mapsto \neg done(\alpha) \rightarrow [\beta]\, V$$

The first reduction rule translates the temporal constraint of α being done before β with an expression in Dynamic Logic that states: *"once action β is performed, it is always the case that action α has been done or otherwise violation V occurs"*. The second reduction rule expresses the violation condition: *"if action α has not been done, once action β is performed it always is the case that violation V occurs"*.

By means of this refinement process, the designer can obtain all the norms and rules that apply in the system, and then include them in the organizational model:

- *role norms* and *role rules* are easily identifiable by the role or group that appears in the norm expression (e.g., see tables 9 and 10).
- *scene norms*, *scene rules*, *transition norms* and *transition rules* come from formulæ expressing precedence relations. If the predicates in a formula refer to landmarks in the same scene, then the formula becomes a scene norm or rule; if the predicates of the formula give a precedence relation between landmarks in different scenes, then the formula becomes a transition norm or rule.

Id	PC_member
Objectives	paper_reviewed(Paper,Report)
Sub-objectives	{read(P), reported(P, Rep), review_received(Org, P, Rep)}
Rights	access-confman-program(*me*)
Norms	O_{PC_member}(*understand(English)*) *done(assign_paper(P,me,Deadline))* $\rightarrow$ O_{PC_member}(*review_paper(P, Rep)* $<$ *do(pass(Deadline))*) *done(assign_paper(P,me,_))* $\wedge$ *is_a_direct_colleague(author(P))* $\rightarrow$ O_{PC_member}(*review_refused(P)* $<$ *pass(TOMORROW)*)
Rules	*done(assign_paper(P,me,Deadline))* $\wedge$ $\neg$*done(review_paper(P, Rep))* $\rightarrow$ [*pass(Deadline)*] *V4* *done(assign_paper(P,me,_))* $\wedge$ *is_a_direct_colleague(author(P))* $\wedge$ $\neg$*done(review_refused(P))* $\rightarrow$ [*pass(TOMORROW)*] *V5*
Type	external

Table 9. Norms and rules in the role description of the *PC member* role.

Group id	Organizers
Roles	{PC-Chair, website manager, general chair, local organizer}
Norms	$F_{organizer}$(*submit_paper(organizer,paper)*)
Rules	[*submit_paper(organizer,paper)*] *V*21

Table 10. Norms and rules in the group description of the *organizer* group.

After refining all the rules that define what the behavior of the system should be, now we focus on the violations that have been identified in previous sections, analyze them and define which are the sanctions. To do so, first we will separate the violations coming from the behavior of external agents (which we call *external violations*) from the ones related to the behavior of the institutional agents (which we call *internal violations*)

Internal violations describe states that the organization should always avoid. As the designer has full control over the design of the agents inside the organization, these agents will fully agree with the objectives of the organization and follow its norms and rules. So, in this case the creation of the *police agent* role and the definition of the violations does not aim to create an *enforcement mechanism* but a continuous *safety control* of the system's behavior (i.e., avoid the system to enter in a non-desirable, *illegal* state because of a failure in one of the agents).

In Omni, *external violations* are the ones where the designer should pay more attention. We cannot assume that agents entering into the organization will always follow the norms and rules imposed by its normative system. Therefore, the internal agents should actively enforce it. As, in our framework, internal agents do not have access to the internal beliefs, goals and intentions of the other agents, they can only check the agents' behavior, by detecting when those agents enter

in states considered *illegal*. The way of doing so is by means of the list of definitions of external violations. Such a list defines, for each violation, the condition that triggers it. This condition is extracted from the rule that defines the violation. As an example, let us take one of the rules identified for the PC member role in table 9:

$$done(paper\_assigned(P,me,Deadline)) \wedge \neg done(review\_paper(P,\ Rep)) \rightarrow [pass(Deadline)]\, V4$$

we can create the condition for violation $V4$ by stating that the action inside [ ] has been *done* [5]. Then we should also add the *sanction* (the actions carried against the violator), the *side-effects* (the actions to be done to counter-act the violation) and the *enforcing roles* (the role or roles that have the responsibility to detect this type of violations):

`Violation:`	$IFMAS{:}V4$
`Pre-conditions:`	$done(assign\_paper(P,me,Deadline)) \wedge \neg done(review\_paper(P,Rep))$ $\wedge done(pass(Deadline))$
`Sanction:`	$delete\_from\_PC\_list(me))$
`Side-effects:`	$\{find\_new\_reviewer(P,r2); assign\_paper(P,r2,Deadline2)\}$
`Enforcing roles:`	$\{organizer, session\_chair\}$

4.3 The Normative Implementation level

There are two main approaches to implement the rules in the rule level:

- Creating a rule interpreter that any agent entering the organization will incorporate.
- Translating the rules into *protocols* to be included in the interaction contracts.

Note that in both cases it is not ensured that the agents will follow those descriptions. The violations in the rule level should also be translated in some detection mechanisms to check the behavior of the agents.

A system where all external agents blindly follow the protocols (as in ISLANDER [13]) is quite efficient from a computational perspective but the agents have only the autonomy to accept/reject the protocol. A system where all external agents have to interpret rules could cope with these problems. However, such a system could not be applied to open environments, as there is a big assumption on the internal architecture of the agents and their way of reasoning.

A better alternative is to be able to accept both kinds of agents. This is the approach taken in the Implementation Level of OMNI, where the organizational dimension provides both the low-level protocols and the related rules. Those (standard) *Autonomous Agents* that are only able to follow protocols, will blindly follow them, while the ones that can also interpret the rules (that is, *Deliberative Normative Agents* [3][7]) will be able to choose among following the protocol or reasoning about the rules, or do both. With this approach the autonomy of the agents entering the organization is adapted to their reasoning capabilities.

In order to allow *Norm Autonomous Agents* to switch from following low-level protocols to higher level rules and norms, there should be a link from procedures to rules, and from rules to norms. An advantage of the OMNI framework is that those links are created by the designer in the process from abstract to concrete norms to rules to the final protocols by means of the successive translations that are made. Those links allow to track, for instance, which abstract norms are related to a given procedure. An example is depicted in figure 10.

4.4 Policies

In our framework policies are elements that group the norms and rules at different levels together depending on the values they fulfill and the objectives they target. Grouping the norms and rules

[5] This translation can be performed, as the predicate *done* complies to the following property: $[\alpha]done(\alpha)$ (*"after action α is performed, it is always the case that α is done"*).

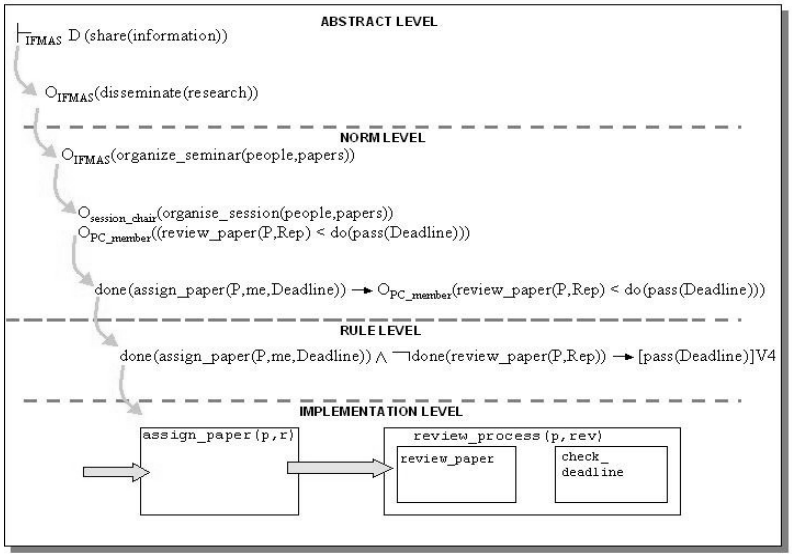

Fig. 10. An example of the refinement process, including the influence of context.

in policies not only eases the development of the normative specification and implementation (as the whole normative specification is modularly divided in policies targeting a given value or objective) but also eases reuse of normative specifications. For instance, in most cases, a *Security policy* is needed, including safety measures, attack avoidance and personal information protection. Although some details in implementation may vary from one MAS to another, most of the abstract and concrete levels of a security policy are similar. Therefore, a (security) policy of a MAS can be reused to define the (security) policy of another MAS.

Policies go from the abstract level (where objectives and values are defined) to the rule level (where those values are described in terms of rules). The resulting specification of the policy is then implemented at the Implementation Level.

5 The Ontological Dimension

The main challenge of coordination and collaboration in open environments is that of mutual understanding. Communication mechanisms include both the representation of domain knowledge (*what* are we talking about) and protocols for communication (*how* are we talking). Both content and protocol have different meanings at the different levels of abstraction (e.g. while at the abstract level one might talk of *disseminate*, such action will most probably not be available to agents acting at the implementation level). Specification of communication content is usually realized using ontologies, which are shared conceptualizations of the terms and predicates in a domain. Agent communication languages (ACLs) are the usual means in MAS to describe communicative actions. ACLs are wrapper languages in the sense that they abstract from the content of communication.

In OMNI, the Ontological Dimension describes both the content and the language for communication, at three different levels of abstraction. At the Abstract Level, the *Model Ontology* can be seen as a meta-ontology that defines all the concepts of the framework itself, such as norms, rules, roles, groups, violations, sanctions and landmarks. Together with the organization's values, these concepts form the basis for communication in the society.

Any mechanism for communication must include both a *knowledge representation language* (to describe knowledge about the domain) and a *communication language* (to specify the interactions among agents). Knowledge representation models are based on ontologies that define the model and vocabulary for a particular domain of discourse. An Agent Communication Language (ACL)

provides the language primitives that enable communication. In the following, will we describe Omni 's approach to both ontologies and communication languages.

5.1 Ontologies

The content aspects of communication, or domain knowledge, are specified by *Domain Ontologies*. The *Concrete Domain Ontology* describes the predicates that are used in the Concrete Level to specify roles, norms and scenes, etc. This ontology includes all the concepts needed for the design of the Organizational and Normative Structure, that is, domain concepts such as goods, participants, roles, locations, time intervals, etc. The complete ontology also includes the specification relationships between concepts, e.g is-a, part-of, synonym, etc.

The *Procedural Domain Ontology*, with the terms from the domain that will be finally used in the implemented system. These concepts are possibly influenced by the specific domain languages used by the agents enacting roles, and may differ between different instantiations of the concrete model. Nevertheless, all concepts at implementation level must translate or implement concepts at the higher levels.

For instance, the actual realization of the AAMAS'04 conference *implements* the IFMAS's objective *organize-conference* defined in the Organizational Model, which in turn *implements* the IFMAS's value of disseminate knowledge, described in its statutes.

5.2 Communication

Communication Acts define the language for communication, including the performatives and the protocols. At the Concrete Level, *Generic Communication Acts* define the interactions languages used in the Organizational Model, while the *Specific Communication Acts* covers the communication languages actually used by the agents as they agree in the interaction contracts. As with the content ontologies, communicative acts defined at a lower level of abstraction implement those defined at a higher level.

In order to allow successful communication, an abstract ACL (Agent Communication Language) is specified in the Abstract Level. As postulated in speech act theory [29] and used in most agent communication languages, agent illocutions are not just propositions that may be true or false, but speech acts that may succeed or fail. Recent work stresses furthermore the importance of relating these communicative primitives to the organizational model of the agent society [32, 30]. In Omni this is represented by the fact that the objectives of a role are related to specific communicative acts (CA) that manipulate concepts of the domain language, defined in the ontology. From a communicative perspective, scene scripts represent abstract conversations, composed by CAs involving the roles that participate in the scene. In the Implementation Level, specific CAs, as used by the agents populating the society, provide the actual communication in the society. Figure 11 depicts this relation between communication at concrete and implementation levels.

Protocols can be seen as conversations between the agents participating in the scene. The communicative abilities of the role enacting agents are the active representation of those CAs. That is, messages which an agent can emit count as the abstract CAs described in the scene script. The scene protocol is then a meaningful combination of messages that achieves the results of the scene and satisfies its norms and constraints.

6 The Agent Perspective

In Omni, "agent" and "role" are fundamentally different concepts. Roles describe the organizational perspective on individuals, whereas agents represent the perspective of the individuals themselves. This difference is not always present in the agent literature where a role often represents a kind of abstract agent or a class of agents. In our work, agents are executable entities, implemented in some language, and have an operational semantics. Roles, on the other hand, are declarative entities meant to represent a part of the organization's design and can be taken up by

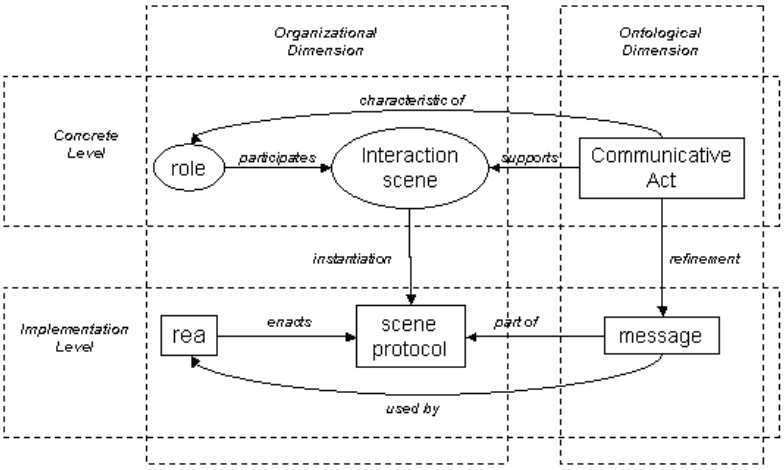

Fig. 11. Linking Organizational and Ontological Dimensions.

the agents enacting the role. So, a role only gets an operational semantics indirectly through the agents that take up that role.

OMNI makes a distinction between two kinds of agents:

- *Institutional agents*, which enact the facilitation roles defined in the Architectural Templates,
- *External agents*, which enact operational roles.

6.1 Institutional agents, facilitation roles and norm enforcement

As explained in §3.2, the Architectural Templates determine the basic facilitation roles necessary for the society. Facilitation roles are usually provided by agents controlled by the society, and follow a trivial contract.[6]

In the case of the *Market* template, facilitation activities of the organization are directed to help participating external agents find suitable partners through identification and matchmaking:

- *Matchmakers* keep track of agents in the system, their needs and possibilities and mediate in the matching of demand and supply of services.
- *Reputation* and *identification* facilities are meant to build the confidence of customers as well as to offer guarantees to society members. Participants can consult a *trusted third party* to request information on the reputation of potential partners.
- *The market master* is responsible for the transactions and to enforce the regulation mechanism that holds in the market.

In the case of the *Network* template, the facilitation agents monitor and help contract formation, take care of introducing (teaching) new agents to the rules of the society and keep track of the reputation of agents. Furthermore, they keep and enforce the norms of the agent community and ensure interaction:

- *Matchmakers*, as in the Market template.
- *Gatekeepers* are responsible for accepting and introducing new agents to the society. Agents entering the system must be informed about the possibilities and capabilities of the society. Gatekeepers negotiate the terms of a social contract between the applicant and the members of the society.

[6] Trivial contracts are agreements that exactly match the role specification, and incorporate no own goals or constraints from the agent.

– *Notaries* keep track of collaboration contracts between agents.
– *Monitoring agents* are trusted third parties. The society will provide monitoring agents to interested parties. When a contract appoints a (set of) monitoring agents, these can check the actions of both parties.

In a *Hierarchy*, the flow of resources or information is coordinated through adjacent steps by controlling and directing it at a higher level in the managerial hierarchy. Facilitation agents are mainly dedicated to the overall control and optimization of the system activities. Sometimes, these facilitation activities are concentrated in one agent, typically the 'root' agent of the hierarchy. Typical roles in hierarchical agent societies are related to control and interface:

– *Root agents* or *controllers* monitor and orient the overall performance of the system (the overall performance) or of a part of it (local performance).
– *Interface agents* are responsible for the communication between the system and the 'outside world'.

Enforcement of norms is one of the tasks of facilitation agents. In Omni norm enforcement is not made in terms of direct control of a central authority over the internal goals or actions that the agents may take, but through the detection of the *violation states* that agents may enter into and the definition, in the social and interaction contracts, of the *sanctions* that are related to the violations. With this approach we do not make strong assumptions about the agents' internal architecture, as the organization will only monitor the agent's behavior (that is, agents are seen as *black boxes*.). The enforcement of the norms in an organization is achieved through a special kind of agents, the *Police Agents*, which monitor the behavior of the agents, detect violations and check the compliance of the sanctions. *Market Masters* and *controllers* are both examples of *Police Agents* in the *Market* and *Hierarchy* templates. *Monitoring agents* are a special case of *Police Agent* in the *Network* template. They only act if both parties request to the society the presence of one or several trusted third parties. This request is included as part of the contract between the parties, and it is the equivalent to the setting up of a (super) contract between the contracting agents and the environment (here personified by the monitoring agents). This super-contract (which can also be described using the contract language) specifies that the monitoring agents are allowed to check the contracting agents actions (e.g., look for violation states) and that the contracting agents must submit to the sanctions imposed.

6.2 The role of external agents

In Omni, an external agent is assumed to be an autonomous, socio-cognitive entity capable of individual social behavior and also assume other agents to have a similar attitude [6]. Socio-cognitive entities are socially able, that is, have the ability to interact and cooperate with others. Omni makes no further assumptions on the internal architecture and design of external agents. This enables the modelling of open agent organizations, where often no knowledge is available about the specific architecture of intervening agents.

Following its own goals and interests, an external agent may seek to enact roles in a society. By joining a society, the agent will take up one or more roles defined in the organizational model. A rational agent will probably choose to enact a role whose objectives somehow contribute to its own goals. We further assume that a rational agent will try to achieve a solution that is as optimal as possible for its own goals, and act as much as possible according to its own characteristics. These may or may not, be different from the expectations the society has of the role. For example, in the conference society, a particular agent that will enact the PC member role may negotiate that she will only review two papers, instead of the 5 society design had in mind for PC members.

In open agent organizations, one of the most important challenges is the specification of mechanisms through which prospective participating agents can evaluate the characteristics and objectives of society roles, in order to decide about their participation. An important aspect concerning role enacting agents is that of modifying the agent to include the characteristics of the assumed role(s). A possible solution for this point has been proposed in [35], which extends agents with an

interface to the society. This interface prevents any action not allowed by the role definition. However, it does not ensure the proactive behavior expected from the role and is not flexible enough to incorporate different enacting styles, capabilities and requirements of the agents. It actually makes the actual agent 'invisible' to the society and only its enactment of the role behavior is apparent in the society. We think that the consequence of an agent adopting a role is more drastic than this. The actual agent behavior must often be modified according to the goals, norms and rights specified by the role.

In the following, we assume that agents have goals of their own, and are able to form (either by design or by deliberation) plans to achieve those roles. OMNI describes agent organizations from a global perspective, rather than from the perspective of the individual agents. Even though agents will take many roles simultaneously and along their life cycles, from the perspective of the society, each rea is a different individual. From the perspective of an OMNI organization it is furthermore up to the agent how to manage and prioritize its goals. That is, by assuming a role, the agent will receive the objectives from that role. How the agent will handle those objectives, whether it interprets them as goals or as norms, what priority it gives them, is up to the agent self. However, the society model is based on the assumption that agents that take up roles are expected to eventually realize the assumed objectives.

Nevertheless, societies are concerned with judging the attitudes of the different reas and how those will affect the performance of the role. That is, the society will not look at the agent as a whole but at how a certain rea is acting. Agent literature extensively discusses different types of social attitudes of agents: selfish, altruistic, honest, dishonest, etc [2] [31] [22] [17] [16]. Different types of agents result in different role performances, because the way an agent will plan its goals, which is dependent on his social attitude, influences the realization of its role objectives and the fulfillment of the role norms. For instance, some types of agents will only attempt to achieve the goals of their adopted roles and forget their own private goals, while others will only attempt to achieve the goals from the role after all their own goals have been satisfied. Furthermore, the relations between agent plans and role objectives, and of agent goals and role sub-objectives must be considered, as well as the influence of the role norms on the behavior of agents. We apply these same concepts to the performance of reas in order to evaluate the different styles of role performance .

Concerning the goals of the agent and the objectives of the role, the following basic types of role enactment by the agents can be distinguished [5]:

1. **Social enactment**: The agent includes as many of its own goals as possible, but gives priority to the objectives of the role over its own.
2. **Maximally social enactment**: The agent only uses the objectives from the role and ignores its own goals, for the duration of the role enactment.
3. **Selfish enactment**: The agent includes as many of its own goals as possible and gives priority to its own goals over the objectives of the role.
4. **Maximally selfish enactment**: The agent only uses its own goals, and ignores any objectives of the role.

From the point of view of the internal agent architecture complexity, in OMNI there is an spectrum of options, from the simplest to the most complex one:

- *Specially tailored agents*, are specifically built to fit completely the proposed agent architecture. One example of this kind of agents are the institutional agents. In the case of the external agents this can only be done for those applications where all the social and interaction contracts are completely pre-defined (including the role enactment and the interaction patterns), so designers may create new agents by only filling in an agent template.
- *Agents using an API to the organization*, can be built independently from the society, which gives more liberty to the development of external agents. The API includes the full definition of the communicative acts for the role enactment and the negotiation of the social and interaction contracts. The agent's interaction with the society is completely fixed by the API.

- *Agents following a loose protocol*, that is, a protocol that is not completely defined. An example of loose protocols in OMNI are the patterns defined in the scene scripts, where only the landmarks are specified. Having agents that are able to follow these patterns gives them more flexibility and more freedom about the exact sequences of actions to be performed to go from one landmark to the next one. This has an impact on the agents' internal architecture, as not only should be able to build plans to go from one landmark to the next, but also check that the execution of these plans complies with the specification of the pattern.
- *Agents following, learning and even reasoning about norms*, that is, agents that can not only follow a (loose) protocol but that can also reason in terms of the related norms and rules, being able to even reason about the consequences of not following a norm. There are two levels of flexibility:
  - *agents breaking the protocols but not the norms*. It is often the case that the implementation of the norms and rules in a protocol is more restrictive than the rules and the norms themselves (for instance, a protocol setting an ordering between two procedures A and B while there are no precedence rules defining precedences between them. A deliberative normative agent could be able, for instance, to detect an unusual delay of the A procedure, see that the B one is running and decide to perform B first. So in this case the agent will break the protocol to cope with the abnormal situation, while keeping its behavior *legal*.
  - *agents breaking the norms*. To violate a norm in order to adhere to a private goal that they consider to be more profitable, possibly including in such consideration the negative value of the repercussions such violation may have. These agents might exhibit the same behavior in most circumstances as the simple agents, but will be able to make better choices in special circumstances. They know when it pays off to violate a rule. As these agents have to reason about the possible consequences of every action, they are quite more complex.

Another important issue in the modelling of open societies is *Ontology Learning*. However, in order to have agents that are able to learn the ontologies used, they should have quite some knowledge about ontologies themselves, about the domain and about the organizational concepts in OMNI (such as roles, rea's, norms, contracts). Although there is some ongoing research on this area [34], it is hard to apply this in practical setups, as it requires the agents to have a huge amount of resources available (e.g., learning capabilities, several huge ontologies).

7 Designing MAS with OMNI

The analysis and design process in OMNI usually results from parallel work in the three dimensions (Organizational, Normative and Ontological), which have been presented in the previous sections. However, depending on the domain of application, one of the dimensions may be more prominent than the others and as such should *guide* the design process.

We can distinguish three kinds of domains:

- **Domains with no norm specification needed**: domains where the normative dimension is not needed as all interactions follow or should follow universally accepted standard patterns of interaction. In this case the norms are implicit on the standard patterns. An example is an auction, where all interaction is defined by game rules, and completely constrained by the organization.
- **Domains with a small normative component**: domains where only some organizational norms are needed in order to support the organizational structure. An example is the *Conference Society* scenario presented in previous sections.
- **Highly-regulated domains**: these are domains where an important body of regulations is to be met, that is, the behavior of the individual agents and/or the system as a whole should follow a set of regulations. Therefore such regulations should be included in the analysis and design process of the MAS. Examples of these kind of domains are applications on law or on other regulated areas such as health or electronic commerce.

In the two domains, the Organizational dimension is the most prominent. In this case the development process is guided by this dimension, with the other two dimensions being developed in parallel accordingly, as explained in previous sections. In the case of highly-regulated domains, the Normative Dimension is the most prominent one and guides the whole process.

In the following section we demonstrate the development of highly regulated domains, by applying OMNI to the design of *CARREL*, a MAS to assist in the organ and tissue allocation process.

7.1 Statutes of the Organization

The statutes of the National Organization for Transplants (ONT) [25] in Spain state the following:

> *The principal objective of the ONT is therefore the promotion of donation and the consequent increase of organs available for transplantation, from which all its other functions result. The ONT acts as a service agency for the whole National Health System, works for the continuing increase in the availability of organs and tissues for transplantation and guarantees the most appropriate and correct distribution, in accordance with the degree of technical knowledge and ethical principles of equity which should prevail in the transplant activity.*

In that statement we can find:

1. the *objectives*: the main objective of this organization is to increase the number of organ donations and the subsequent increase in available organs for transplants. Another objective is to properly allocate organs.
2. the *context*: ONT states that it operates inside the Spanish National Health System, and such statement clearly defines the context of the organization.
3. the *values*: The latter part indicates the values according to which the ONT operates. The values are the following: to guarantee the most *appropriate*, *correct* and *equal* distribution of pieces among potential recipients. Where *appropriate*, *correct* and *equal* are vague terms defined by both technical (medical) and ethical standards. Implicitly it also says that ethical values are also part of the organization. These values will also play a role in the regulations that will determine the actual process according to which the transplantation should be performed.

7.2 Norms and Rules, Roles

The next step is to describe the values of the organization in terms of desires:

$$ONT{:}D1 \vdash_{ONT} D(\mathit{appropriate}(\mathit{distribution}))$$
$$ONT{:}D2 \vdash_{ONT} D(\mathit{correct}(\mathit{distribution}))$$
$$ONT{:}D3 \vdash_{ONT} D(\mathit{equal}(\mathit{distribution}))$$

It is important to note here that the context of the organization may affect its specification at all levels. During the process we will continuously check if these super-contexts may define a restriction that affects the ongoing specification. different levels. In this case, the context of ONT is the Spanish National Health system (SpNHS), The Spanish Law (SpLaw) and the European Law (EULaw). At the level of values, the SpNHS defines some values[7] that are inherited by the ONT: [8]

[7] Notation: for values, norms and rules coming from an article in a given law or regulation, we will use the notation *law* : *article*. For instance, RD.2070.99:5 means *Article 5 in Royal Decree 2070/99*, while LGS:3.2 means *Article 3.2 in LGS* (Generic Health Law).

[8] Note here the use of the *contextual translation* operator at the level of values. It is used to say that a value (such as ONT:D4.1) *translates* (in the ONT context) the value expressed in RD.2070.99:5. (i.e., ONT:D4.1 fixes the interpretation of RD.2070.99:5 in the ONT context). It is also important to note here that $ONT{:}D3.1 \rightsquigarrow_{ONT} ONT{:}D3$ and $ONT{:}D3.2 \rightsquigarrow_{ONT} ONT{:}D3$. In order to be concise, we will not add these relations, that can be inferred by the naming convention we have chosen.

$$ONT{:}D4.1 \vdash_{ONT} D(\textit{anonymity}(\textit{donor})) \rightsquigarrow_{ONT} RD.2070.99{:}5$$
$$ONT{:}D4.2 \vdash_{ONT} D(\textit{anonymity}(\textit{recipient})) \rightsquigarrow_{ONT} RD.2070.99{:}5$$

These values state that anonymity of all patients should be kept. These values were not covered by any ONT value. Therefore, they should be added to the list of ONT values. But in some cases context refines some of the values:

$$ONT{:}D3.1 \vdash_{ONT} D(\textit{no\_discrimination}(\textit{recipient, race})) \rightsquigarrow_{ONT} LGS{:}10.1a$$
$$ONT{:}D3.2 \vdash_{ONT} D(\textit{no\_discrimination}(\textit{recipient, sex})) \rightsquigarrow_{ONT} LGS{:}10.1b$$
$$ONT{:}D3.3 \vdash_{ONT} D(\textit{no\_discrimination}(\textit{recipient, ideology})) \rightsquigarrow_{ONT} LGS{:}10.1c$$
$$ONT{:}D3.4 \vdash_{ONT} D(\textit{no\_money}(\textit{donor, piece})) \rightsquigarrow_{ONT} RD.2070.99{:}8$$
$$ONT{:}D3.5 \vdash_{ONT} D(\textit{no\_money}(\textit{recipient, piece})) \rightsquigarrow_{ONT} RD.2070.99{:}8$$

The organizational values above express that no patient should be discriminated on the basis of race, sex, ideology or social status, and that organ and tissue transplantation should be a free service. In these case these are not new values, but refinements of the vague concept of *equity* described in value $ONT{:}D3$.

At the level of values we can already identify the roots of the different policies of the organization:

- the *organ allocation* and *tissue allocation policies* have its roots on values ONT:D1, ONT:D2, ONT:D3.1, ONT:D3.2, ONT:D3.3, ONT:D3.4 and ONT:D3.5;
- the *security policy* has its roots on values ONT:D2, ONT:D4.1 and ONT:D4.2.

In the sake of simplicity we will focus only in the organ allocation policy, and more concretely in the appropriate distribution value ($ONT{:}D1$).

As we saw in §4.1, desires are then translated into abstract norms. In this case $ONT{:}D1$ value can be translated by a single abstract norm as follows:[9]

$$ONT{:}A1\ O_{ONT}(\textit{appropriate}(\textit{distribution})) \mapsto_{ONT} ONT{:}D1$$

Again in this case context has influence in the refining process. Apart of adding new norms, Spanish Law by Article 19 in Royal Decree 2070/99 states that the quality and safety of pieces (organs and tissues) should be ensured. In this case this norm is related with the appropriateness of pieces, so this norm is refining ONT:A1. The result of introducing this norm is the following:

$$ONT{:}A1.1\ O_{ONT}(\textit{ensure\_quality}(\textit{piece})) \rightsquigarrow_{ONT} RD.2070.99{:}19b$$
$$ONT{:}A1.2\ O_{ONT}(\textit{ensure\_safetiness}(\textit{piece})) \rightsquigarrow_{ONT} RD.2070.99{:}19b$$

Once the Abstract Norms are identified, the next step is to refine them in more Concrete Norms:

[9] Note that in this case we have not used the *contextual translation* operator ($\rightsquigarrow$), as, in this case, we have translated from the language of values to the language of abstract norms. Therefore, we use a language translation operator $\mapsto$ which allows us to keep track of the transition.

$ONT{:}C1.1$ $O_{ONT}(ensure\_quality(piece)) \equiv ONT{:}A1.1$
$ONT{:}C1.1.1$ $O_{ONT}(ensure\_quality(piece) < do(assign(piece,recipient)))$
$ONT{:}C1.1.1.1$ $O_{origin\_org}(ensure\_quality(piece))$
$ONT{:}C1.1.1.1.1$ $O_{hospital}(ensure\_quality(organ))$
$ONT{:}C1.1.1.1.2$ $O_{tissue\_bank}(ensure\_quality(tissue))$
$ONT{:}C1.1.1.2$ $O_{CARREL}(get\_quality(piece,origin) < do(assign(piece,recipient)))$
$ONT{:}C1.1.1.2.1$ $O_{CARREL}(get\_quality(organ,hospital) < do(assign(organ,recipient)))$
$ONT{:}C1.2$ $O_{ONT}(ensure\_safetiness(piece)) \equiv ONT{:}A1.2$
$ONT{:}C1.2.1$ $O_{ONT}(ensure\_compatibility(piece,\ recipient) < do(assign(piece,\ recipient)))$
$ONT{:}C1.2.1.1$ $O_{CARREL}(ensure\_compatibility(piece,\ recipient) < do(assign(piece,\ recipient)))$
$ONT{:}C1.2.1.1.1$ $O_{CARREL}(ensure\_compatibility(organ,\ recipient) < do(assign(organ,\ recipient)))$

ONT:C1.1 and ONT:C1.2 are just copies of ONT:A1.1 and ONT:A1.2. In this first set of norms about quality and the compatibility of the pieces are refined. In the case of norm ONT:C1.1.1, first we have introduced a time constraint (quality assessment should be done before the piece is assigned). Then a separation of duties is made among roles. For the organizations that are originators of the piece, as they do not perform the assignation, they only have the obligation to ensure quality of pieces, while for the *CARREL* MAS the quality assessment is reduced to get such information from the organization (hospital or tissue bank) which is the origin of the piece. The information about the quality of piece should be catched before the assignation is made, introducing here a restriction to reduce temporal abstractness. A similar process is made for obligation ONT:C1.2. In this case the obligation to ensure compatibility only lays on *CARREL*, as it is the one performing the assignation.

This reduction on the role abstraction goes in parallel with the definition of the role hierarchy. The process starts with a root node, and then it distributes the objectives among roles. The result of this process can be seen in figure 12. At the root node we find that *patients* have no contribution to the *distribute organs* objective, so in this case the objective passes unchanged to the *health organization* role. Then we distribute the objective between the *ONT* (which manages the requests and offers for organs) and the *hospitals* (which may request or offer organs and also have to coordinate the transplantation process inside the hospital). To finalize the distribution, the *manage organ requests and offers* objective is mainly given to the *CARREL* MAS, but ONT's *board* has also a contribution as it has to monitor the process.

The next step is to express the norms in terms of operational rules. In the case of the norms above, the translation is as follows[10]:

$CRL{:}R1.1$ $[assign(organ,\ recipient)]done(get\_quality(organ,\ hospital)) \mapsto_{CRL} ONT{:}C1.1.1.2.1$
$CRL{:}R1.2$ $\neg done(get\_quality(organ,\ hospital) \rightarrow [assign(organ,\ recipient)]CRL{:}V1 \mapsto_{CRL} ONT{:}C1.1.1.2.1$
$CRL{:}R3.1$ $[assign(organ,\ recipient)]done(ensure\_compatibility(organ,\ recipient)) \mapsto_{CRL} ONT{:}C1.2.1.1.1$
$CRL{:}R3.2$ $\neg done(ensure\_compatibility(organ,\ recipient)) \rightarrow [assign(organ,\ recipient)]CRL{:}V3 \mapsto_{CRL} ONT{:}C1.2.1.1.1$

This set of rules can be further refined by adding domain knowledge. E.g. $ensure\_compatibility(organ, recipient))$ is defined in terms of donor-recipient compatibility rules[11]:

[10] Notation: All violations are identified by a code following the format $CRL : Vx$.
[11] These rules for the case of kidneys are a subset of the ones presented in [36].

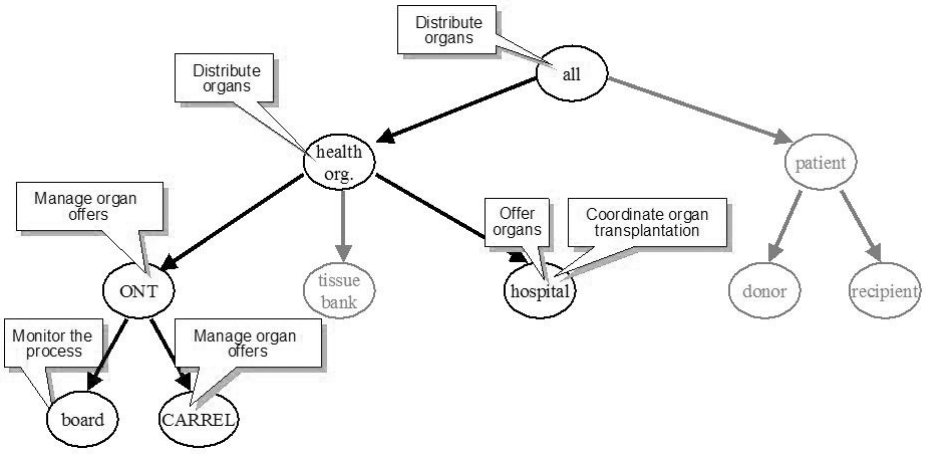

Fig. 12. Role hierarchy resulting from the distribution of the *distribute organs* goal.

```
1- (age_donor >= 60) AND (age_donor < 74) AND (creatinine_clearance > 55 ml/min)
     -> (age_recipient >= 60) AND (transplant_type SINGLE-KIDNEY)

2- (age_donor >= 60) AND (age_donor < 74) AND (glomerulosclerosis <= 15%)
     -> (age_recipient >= 60) AND (transplant_type SINGLE-KIDNEY)
```

At this point we should also consider rules that context may impose on the allocation process. In this case none of the super-contexts of ONT imposes rules, as Spanish Law gives ONT the full power to rule in the allocation process. However, there are some rules ONT has defined and are important to be introduced here. One of them is about the precedence of urgency_0 patients (i.e., patients in critical condition) over any other patient:

$$(compatible(patient_1, piece) \wedge compatible(patient_2, piece) \\ \wedge urgency\_0(patient_1)) \Rightarrow assign(patient_1, piece)$$

With such a rule we express that, given two patients that are compatible with a given piece, if one of them has the urgency_0 level then the piece should be assigned to the urgency_0 patient. We can express this rule as follows:

$$CRL\!:\!R16.1\ [assign(organ,\ patient)]done(check\_if\_urgency\_0(organ)) \rightsquigarrow_{CRL} ONT\!:\!R1$$
$$CRL\!:\!R16.2\ \neg done(check\_if\_urgency\_0(organ)) \rightarrow [assign(organ,\ patient)]CRL\!:\!V16 \\ \rightsquigarrow_{CRL} ONT\!:\!R1$$
$$CRL\!:\!R17.1\ [assign(tissue,\ patient)]done(check\_if\_urgency\_0(tissue)) \rightsquigarrow_{CRL} ONT\!:\!R1$$
$$CRL\!:\!R17.2\ \neg done(check\_if\_urgency\_0(tissue)) \rightarrow [assign(tissue,\ patient)]CRL\!:\!V17 \\ \rightsquigarrow_{CRL} ONT\!:\!R1$$

Note here the use of the *contextual translation* operator to express that these rule are interpretations (or implementations) of the above-mentioned ONT's rule (which we have tagged as ONT:R1).[12]

A unwritten rule that ONT is trying to implement is that hospitals should return information about the result of the transplantation from each piece, in order to record the results of the implant and also to allow ONT to detect any weak point in the allocation process to be repaired.[13] We can express such a rule as follows:

[12] In the examples of this section we will see again the use of the $\rightsquigarrow_s$ operator referring to external rules. In all those cases we assume that those external rules are also formalized in terms of Dynamic Logic.

[13] Now a days, ONT only keeps a nation-wide registry about liver transplantation. On the other hand, OCATT already keeps registries about heart, liver, kidney and stem cells.

$CRL{:}R18.1$ $[deliver(organ,\ hospital_1,\ hospital_2)](is\_time(t)$
$\rightarrow [wait\_for(is\_time(t\ +\ 1\ day))]done(hospital_2.send(result\_transplant)))$
$\rightsquigarrow_{CRL} ONT{:}R2$
$CRL{:}R18.2$ $(done(deliver(organ,\ hospital_1,\ hospital_2)) \wedge is\_time(t)$
$\wedge \neg done(hospital_2.send(result\_transplant)))$
$\rightarrow [wait\_for(is\_time(t\ +\ 1\ day))]CRL{:}V18.1 \rightsquigarrow_{CRL} ONT{:}R2$

The first rule expresses the following: *"once the organ is sent, if* t *is the current time then when time* t + *1 day has passed it always should be that the hospital has sent the data about the transplantation"*. The second is the violation rule, which expresses the following: *"if the organ has been sent at time* t *and the hospital has not sent the data about the transplantation, once time (*t + 1 day*) passes the violation CRL:V18.1 is triggered"*. The same structure can be used to express the evolution reports three months later and a year later.

At this point we have all the components needed to define a violation at the Rule Level. Following there is an example of the complete definition of the CRL:V18.1 violation:

`Violation:`	$CRL{:}V18.1$
`Pre-conditions:`	$(done(deliver(organ,\ hospital_1,\ hospital_2)) \wedge is\_time(t)$
	$\wedge done(wait\_for(is\_time(t\ +\ 1\ day)))$
	$\wedge \neg done(hospital_2.send(result\_transplant)))$
`Sanction:`	$\{request(hospital_2,\ send(organ,\ result\_transplant));$
	$inform(board,$`precondition`$)\}$
`Side-effects:`	$\{record(\neg done($`precondition`$),incident\_log)\}$

7.3 Organizational Model

Once the set of norms, rules and violations has been defined, the next step is to define the organizational structure that will give support to them, using the objectives in the statutes as an starting point but also taking into account the decisions taken in the previous step. In order to have a guide in the design process, we should first identify the architectural template(s) suitable for this application [12]. In this case, on the one hand, this application should support the collaboration of several, distributed entities (the hospitals and tissue banks) in order to reach a common objective (a suitable allocation of human organs and tissues for transplantation purposes). The *Network* template is well-suited for this. On the other hand the behavior of the system should be, by law, controlled and monitored at all times, and there exist authorities that regulate the behavior of the system as a whole and of the individual actors. The *Hierarchy* template is well-suited for this. Therefore in this application we should combine both the Network and the Hierarchy templates.

Once the architectural templates have been identified, the next step if to define the Social Structure. This means to identify the roles to be enacted by software agents and the relations between those roles. In the case of the *distribute organs* objective, we will use as starting point the objective distribution already defined in figure 12. We should refine those roles that are too abstract to be enacted by humans or by a single agent. One of such roles is the *CARREL* role. This role represents the whole agent-mediated organization that will mediate in the organ and tissue allocation. The *CARREL* role has, as assigned objective, to manage organ offers. This is a facilitation role, but too abstract. As a guidance for the refinement of this role in more concrete ones, we can use the facilitation roles defined by the *Network* template (*Matchmaker*, *Gatekeeper*, *Notary* and *Monitor*) and the *Hierarchy* template (*Controller* and *Interface facilitator*). The result is the following role list:

- *admission*: to authenticate the agents coming from hospitals to offer or request tissues. Therefore, is the role related with access control to the system. It is an instance of the *Gatekeeper* role.
- *allocator*: to assign organs, ensuring compatibility between donor and recipient. It is an instance of the *Matchmaker* role.

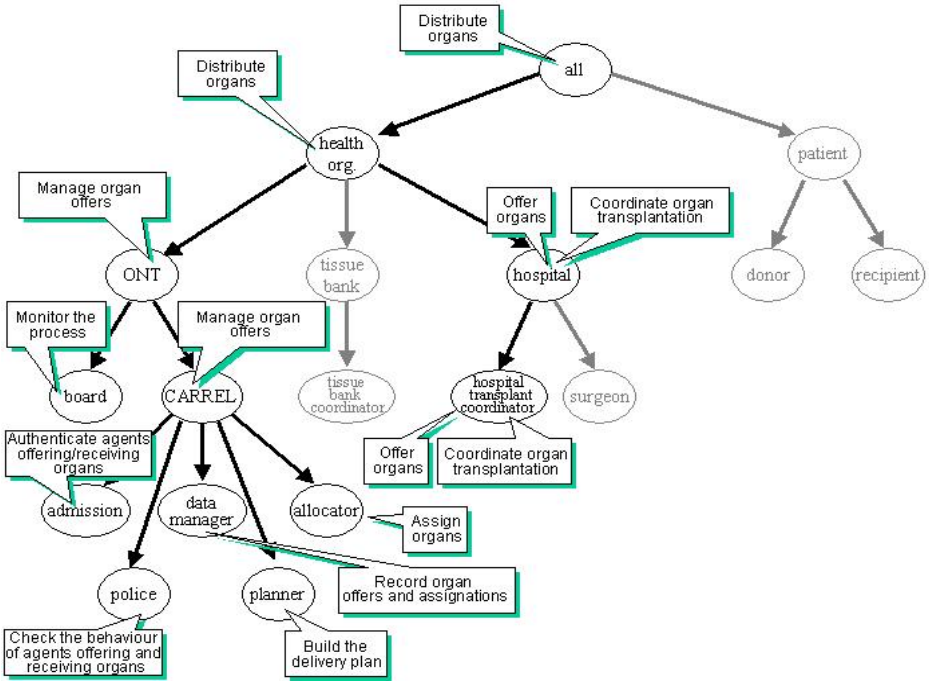

Fig. 13. Extension of the Role hierarchy for the *distribute organs* goal.

- *data manager*: providing an interface to the systems' database to ensure (role based) access control to the data stored in the database, and to register all the events that occur inside the organization. It is an instance of both the *Notary* and the *Interface facilitator* roles.
- *police*: to detect when the behavior of the agents inside *CARREL* may create a violation state and then act properly (i.e., execute the defined *sanctions* and *side-effects* for that violation). It is an instance of the *Monitor* role.
- *CARREL* : as an instance of the *Controller* role, we keep the *CARREL* role to monitor and guide the behavior of the system as a whole.
- *planner*: to build the delivery plan (from a hospital to another hospital). It is not an instance of a facilitation role in the architectural templates, but a facilitation role specifically needed in this application.

We also have to refine the other roles in the role hierarchy involved in the *distribute organs* objective. In the case of the *hospitals*, the assigned objective is to be carried out by the (*hospital transplant coordination* role). The surgeon role does not intervene here but in tissue allocation. The resulting extended hierarchy is shown in figure 13.

Then the role dependencies between these roles should be defined. The resulting diagram is depicted in figure 14. It shows, for instance, how the *allocator* role needs 1) the *admission* role to certify the identity of the agents in order to do the assignation, and 2) the *planner* role to know if delivery of a given organ to a given patient is feasible given the time constraints imposed by the organ itself.

Once all relations between roles have been defined, then each role has to be properly described, including its objectives, rights, norms and rules. For instance, table 11 shows the description of the *allocator* role, including the norms and rules that apply to it, concerning all the checks that have to be done before assigning an organ or a tissue. Table 12 is an example of description of a role meant to be enacted by external agents. In this case the *hospital_transplant_coordinator* represents the hospital, it may offer or request organs and keeps updated the *CARREL* system about the organs that are available for transplantation. The norms that apply to this role are

Id	allocator
Objectives	{assign(organ,patient), assign(tissue,patient)}
Sub-objectives	{check_if_urgency0(organ), ensure_quality(organ), ensure_compatibility(organ, recipient), ...}
Rights	{access-to-database(allocator, DB-organs), access-to-database(allocator, DB-tissues)}
Norms	$O_{allocator}(ensure\_quality(organ) < do(assign(organ,\ recipient)))$ $O_{allocator}(ensure\_quality(tissue) < do(assign(tissue,\ recipient)))$ $O_{allocator}(ensure\_compatibility(organ,\ recipient) < do(assign(organ,\ recipient)))$ $O_{allocator}(ensure\_compatibility(tissue,\ recipient) < do(assign(tissue,\ recipient)))$
Rules	$[assign(organ,\ patient)]done(check\_if\_urgency\_0(organ))$ $[assign(organ,\ recipient)]done(get\_quality(organ,\ hospital))$ $[assign(tissue,\ recipient)]done(get\_quality(tissue,\ tissue\_bank))$ $[assign(organ,\ recipient)]done(check\_compatibility(organ,\ hospital))$ $[assign(tissue,\ recipient)]done(check\_compatibility(tissue,\ tissue\_bank))$
Type	internal

Table 11. *allocator* role description.

Id	hospital_transplant_coordinator
Objectives	{get(organ,patient), get(tissue,patient), offer(organ,$CARREL$)}
Sub-objectives	{manage_waiting_list(organs), check_quality(organ), ...}
Rights	update-database(hospital_transplant_coordinator, DB-organs)
Norms	$O_{hospital\_transplant\_coordinator}(ensure\_accuracy(patient,\ data))$ $O_{hospital\_transplant\_coordinator}(ensure\_security(patient,\ data))$
Rules	$[send(patient\_data,CARREL)]done(avoid\_possible\_attack(patient\_data))$ $[deliver(organ,\ hospital_1,\ hospital_2)](is\_time(t)$ $\rightarrow [wait\_for(is\_time(t + 1\ day))]done(send(result\_transplant,CARREL)))$ $[deliver(organ,\ hospital_1,\ hospital_2)](is\_time(t)$ $\rightarrow [wait\_for(is\_time(t + 3\ months))]done(send(result\_transplant,CARREL)))$ $[deliver(organ,\ hospital_1,\ hospital_2)](is\_time(t)$ $\rightarrow [wait\_for(is\_time(t + 1\ year))]done(send(result\_transplant,CARREL)))$
Type	external

Table 12. *hospital_transplant_coordinator* role description.

Group id	internal_roles
Roles	{$CARREL$, allocator, police, admission, data_manager, planner}
Norms	$O_{internal\_roles}(ensure\_security(patient,\ data))$
Rules	$[input(external\_roles,\ data)]done(check\_access\_rights(external\_roles,\ data))$ $[query(external\_roles,\ data)]done(check\_access\_rights(external\_roles,\ data))$

Table 13. *internal_roles* group description.

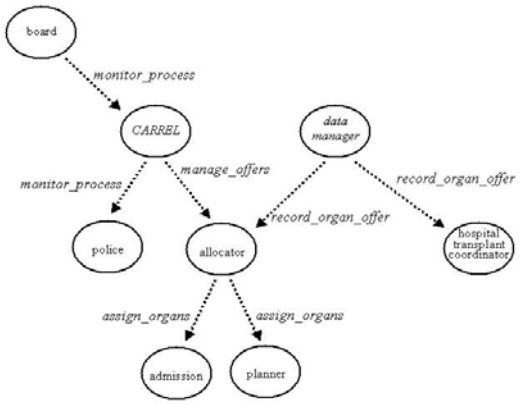

Fig. 14. An example of role dependencies in the organ allocation scenario.

mainly about the accuracy and security of the data about patients that sends to *CARREL*, while the rules define a process to keep the information about the success of the transplantation up to date and the health state of recipients (one day, three months and one year after the organ was delivered to the hospital).

To end the *social structure* definition, groups may be defined. Table 13 is an example of group description in the organ allocation scenario. It groups together all the internal roles, as they all share some common norms and rules that they have to comply with. The *internal_roles* description refers to *external_roles*, which is another group composed by the *hospital_transplant_coordinator* and *tissue_bank_coordinator* roles which external agents entering into the system may enact.

The process to define the *interaction structure* is equivalent to the one described in §3.2. Figure 15 depicts the interaction structure, composed by 5 scenes: *Admission*, where external agents enter and have to identify themselves; the *Organ* and *Tissue Exchange* scenes, where the search for suitable organ recipients or tissues is done, the *Consultation* scene, where external agents can query or update the information about waiting lists, organs offered, tissues and transplantation results; and the *Exchange Confirmation* scene, where the final assignment is confirmed by ensuring that there is a suitable delivery plan, that the organ or tissue has the proper quality and that the hospital is prepared to receive the organ or tissue.

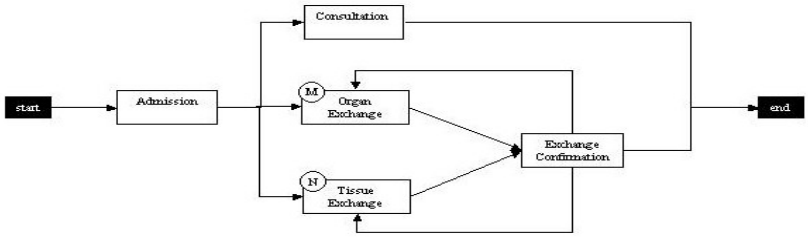

Fig. 15. Interaction Structure in the organ allocation scenario.

While in the conference scenario the landmarks and interaction patterns are completely defined by the designer, in normative environments (such as the organ allocation) most of the landmarks and the interaction patterns can be extracted from the precedence rules already identified. A good example are the rules that apply to the interaction in the *admission* scene:

$CRL: R31.1\ [access(external\_roles,\ CARREL)]done(check\_identity(external\_roles))$
$CRL: R32.1\ [input(external\_roles, data)]done(check\_access\_rights(external\_roles, data))$
$CRL: R38\quad [input(external\_roles,\ data)]access(external\_roles,\ CARREL)$

These rules defines tree precedence relations between four actions (*check_identity* BEFORE *access*, *access* BEFORE *input*, *check_access_rights* BEFORE *input*). As the fulfilment of these actions are important steps of the interaction, we can create four landmarks ($L1$ to $L4$) representing the states where these actions have been done:

$L1\ DONE(admission, check\_identity(agent))$
$L2\ DONE(admission, check\_access\_rights(agent, data))$
$L3\ DONE(external\_roles, access(external\_roles, CARREL))$
$L4\ DONE(external\_roles, input(external\_roles, request\_data))$
$L5\ DONE(admission, accurate(request\_data))$

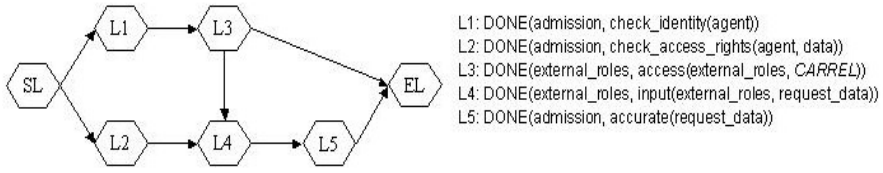

Fig. 16. Landmarks for the Admission scene in the organ allocation scenario.

As a design decision we also add an extra landmark ($L5$), to check that the information in the request coming from the external agent is correct. As a result, the scene script can be represented graphically (figure 16) and the scene completely defined (table 14). A similar process can be followed for the other scenes, to complete the interaction structure definition.

Scene	Admission
Roles	admission (1),external_roles (2..Max)
Results	$r_1 = \forall agent : enacts(agent, external\_roles) :$ $identity\_checked(agent) \wedge request\_checked(agent, request)$
Interaction Patterns	$PATTERN(r_1)=$ {DONE(admission, $check\_identity(agent)$), DONE(admission, $check\_access\_rights(agent, data)$), DONE(external_roles, $access(external\_roles, CARREL)$), DONE(external_roles, $input(external\_roles, request\_data)$), DONE(admission, $accurate(request\_data)$)}
Norms	$O_{admission}(ensure\_security(donor,\ data))$ $O_{admission}(ensure\_security(recipient,\ data))$
Rules	$[access(external\_roles,\ CARREL)]done(check\_identity(external\_roles))$ $[input(external\_roles, data)]done(check\_access\_rights(external\_roles, data))$ $[input(external\_roles,\ data)]access(external\_roles,\ CARREL)$

Table 14. Script for the *Admission* scene.

7.4 Organizational and Normative Implementation

The implementation level focus on the agents that will (a) enact the roles, (b) follow the interaction structures and (c) meet the norms and rules, all of them defined in the Concrete Level.

Therefore, as explained in §3.3, the first step is to specify the *role enacting agents* (*rea's*). In the case of the internal roles, as they are essential for the proper behavior of the society, it is usually the case that the agents that will enact these roles will be created to fulfill completely the role(s) specification. Therefore in this case it is enough to have a trivial social contract, with no additional clauses[14] (see table 15). In the case of external roles, there will be some times that additional clauses may be introduced. An example is shown in table 16, where an external agent enacting the *hospital_transplant_coordinator* role changes the deadlines for recipient state updates, from (1 day, 3 months, a year) to (3 days, 2 months, a year).

Agent alias	Alloc-1
Agent IOR	000000000101000000000E3539272E35436E32345E322...
Role	*allocator*
Contract Clauses	{ }

Table 15. Social contract for agent *Alloc-1*.

Agent alias	Sant_Pau_Coord
Agent IOR	000000000101000000000E317564E34532E39675E331...
Role	*hospital_transplant_coordinator*
Contract Clauses	{[*deliver*(*organ, hospital*$_1$, *hospital*$_2$)](*is_time*(*t*) $\rightarrow$ [*wait_for*(*is_time*(*t + 3 days*))]*done*(*send*(*result_transplant*,*CARREL*))), [*deliver*(*organ, hospital*$_1$, *hospital*$_2$)](*is_time*(*t*) $\rightarrow$ [*wait_for*(*is_time*(*t + 2 months*))]*done*(*send*(*result_transplant*,*CARREL*))), [*deliver*(*organ, hospital*$_1$, *hospital*$_2$)](*is_time*(*t*) $\rightarrow$ [*wait_for*(*is_time*(*t + 1 year*))]*done*(*send*(*result_transplant*,*CARREL*))) }

Table 16. Social contract for agent *hospital_transplant_coordinator*.

The *Interaction Model* results from the instantiation of the scene scripts defined in the Concrete Level, to the specific circumstances of the rea's. Such scenes are defined by *interaction contracts*. Table 17 gives an example of the interaction contract that defines the *Admission* scene. The contract specifies that agent *Adms1* and agent *Sant_Pau_Coord* will interact following the protocol *P-admission1* (shown in figure 17). In this case no extra clauses are introduced in the interaction.

Agent aliases and roles	{Adms1 (*admission*), Sant_Pau_Coord (*hospital_transplant_coordinator*)}
Scene	*Admission*
Contract clauses	{ }
Protocol	P-admission1

Table 17. Example of an interaction contract.

[14] Even in the case that an external agent is allowed to enact an internal role, it will be case that this agent has to fully comply with the role specification, with no added clauses.

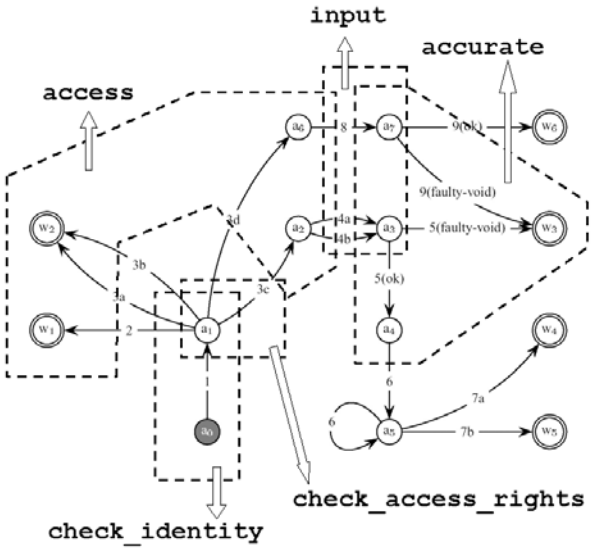

Fig. 17. Relation between the landmarks and the protocol *P-admission1* in the *Admission* scene.

It is important to note here that the final protocols should comply with the landmarks defining the interactions patterns of an scene. The precise mechanisms for automatic negotiation of the protocols by the agents and the model checkers needed to check protocol compliance of scene's interactions patterns are still ongoing work. At this point the interaction patterns are used by the designer as a guideline to define the protocol. Figure 17 shows an example, where we used the landmarks in figure 16 as a basis to define the final scene protocol (using ISLANDER [13]).

8 Discussion: OMNI compared with other approaches

Development methods for multi-agent systems are currently a hot research topic and several approaches have been proposed. Comprehensive methodologies to design agent societies must be able to describe the characteristics of organizational environments. Such environments are best understood in terms of social concepts such as organization structures, norms and domain language. Furthermore, methodologies must support the development of open societies and the specification of formal institutions. These are issues covered by the OMNI framework. In the remainder of this section, we briefly discuss how some well known models support the social and normative concepts introduced by the OMNI framework.

GAIA Gaia [40] is one of the first agent-oriented software engineering methodologies that explicitly takes into account social concepts. Gaia aims at providing a coherent conceptual framework for the analysis and design of multi-agent systems. The analysis phase results in 1) the agent model which specifies system roles and their characteristics in terms of permissions (the right to exploit a resource) and responsibilities (functionalities); and, 2) the interaction model, which captures the dependencies and relations between roles by means of protocol definitions.
Discussion. Gaia models are situated in at the Concrete Level of OMNI (cf. figure 1). While the implementation Level is explicitly and purposefully ignored, Gaia does not have any notion of an Abstract Level. Gaia is only concerned with the society level and does not capture internal aspects of agent design. However, societies are only considered from the perspective of the individual participants, and therefore Gaia does not deal with communication or other collective issues. Furthermore, normative aspects are reduced to static permissions, a sort of constraints or rules

and behavior is fixed in protocols. Moreover, Gaia is not suited to model open domains, and cannot easily deal with self-interested agents, as it does not distinguish between organizational and individual aspects, and does not provide capabilities for agent interpretation of society objectives, norms or plans.

SODA SODA [24], is actually an extension to Gaia that enables open societies to be designed around suitably-designed coordination media, and social rules to be designed and enforced in terms of coordination rules. As Gaia, SODA distinguishes between an analysis and a design phase. The analysis phase results in three different models: the *role model*, describe the goals, or tasks of roles and groups; the *resource model* that described the environment in terms of available resources; and the *interaction model*, where interaction protocols describe the information required and provided by roles and resources, and the rules governing interaction. During the design phase, roles are mapped to agent classes (the *agent model*), groups are mapped into societies designed around coordination abstractions (the *society model*), and resources are mapped into infrastructure classes (the *environment model*).
Discussion. As an attempt to include an higher abstraction level (cf. figure 1), SODA presents a notion of the context, or environment, of the society, albeit not explicit. However, even though SODA distinguishes between agent and collective spaces, it sees roles as the representation of the observable behavior of agents, and therefore cannot represent the difference between the organizational perspective on the activity and aims of individuals (represented by the concept of role in Omni) from the agent perspective on its own activity and aims (represented by the concept of agent in Omni and linked to the role by a social contract). Role enactment is fixed in SODA as the agent model that maps roles to agent classes without any possibility to accommodate agent preferences or characteristics (agent classes are pure specifications of the role characteristics). There are no normative aspects in SODA further than the notion of permission to access infrastructure services. Communication primitives are limited to interaction protocols, and SODA provides no explicit representation for the domain ontology. Furthermore, SODA also does not have a clear and formal semantics.

ISLANDER The ISLANDER formalism [13] provides a formal framework for institutions [23, 28] and has proven to be well-suited to model practical applications (e.g. electronic auction houses). This formalism views an agent-based institution as a *dialogical system* where all the interactions inside the institution are a composition of multiple dialogic activities (message exchanges). These interactions (or *illocutions* [23]) are structured through agent group meetings called *scenes* that follow well-defined protocols. This division of all the possible interaction among agents in scenes allows a modular design of the system, following the idea of other software modular design methodologies such as the Modular Programming or Object Oriented Programming. A second key element of the ISLANDER formalism is the notion of agent's *role*. Each agent can be associated to one or more roles, and these roles define the scenes the agent can enter and the protocols it should follow. Finally, this formalism defines a graphical notation that not only allows to obtain visual representations of scenes and protocols but is also very helpful while developing the final system, as they can be seen as blueprints. ISLANDER has been mainly used in *e*-commerce scenarios, and was used to model and implement an electronic Auction house (the *Fishmarket*). Furthermore, the e-INSTITUTOR platform and the ISLANDER API enable the animation of models and the participation of external agents. The activity of these agents is, however, constrained by *governors* that regulate agent actions, to the precise enactment of the roles specified in the institution model.
Discussion. In contrary to the other frameworks discussed here, ISLANDER provides a sound model for the domain ontology and has a formal semantics [23, 28]. However, ISLANDER provides no primitives to model the Abstract Level of an organization and does not consider the normative aspects of organizations, further than the specification of constraints for scene transition and enactment (the only allowed interactions are those explicitly represented by arcs in scenes).

TROPOS TROPOS methodology [1] [4] spans the overall development process. It distinguishes between an early and a late requirements phase, and between architectural design and detailed design . Tropos starts with an analysis of the organizational setting of the application, in the early requirements phase. The strategic dependency model describes an 'agreement' between two actors: the depender and the dependee. The strategic rationale model determines through a means-ends analysis how an actor's goals (including soft goals) can actually be fulfilled through the contributions of other agents. These two models serve as input for the late requirements phase where a list of functional and non-functional requirements is specified. The architectural design defines the structure of a system in terms of subsystems that are interconnected through data, control and other dependencies. The detailed design defines the behavior of each component. The models are implemented using Jack Intelligent Agents [15], which is an agent-oriented extension of Java.
Discussion. Tropos is a fairly complete methodology that considers all steps in system development, and it treats both inter-agent and intra-agent perspectives. The early requirement phase of Tropos, can be seen as a specification of the Abstract Level proposed by OMNI (cf. figure 1). The late requirements phase comes fairly close to the idea of Concrete Level in OMNI , except that it does not provide explicit concepts to capture norms, and ontological aspects are only implicitly described. At the Implementation Level, Tropos provides a detailed implementation of organizational models into JACK agents. The main two shortcomings of Tropos are that a) it is not formal (although there is some ongoing work on providing a formal semantics for Tropos), and b) it is too organizational-centered in the sense that is does not consider that agents can have their own goals and plans, and not just those coming from the organization. Furthermore, Tropos has no concept representing the normative aspects of an organization.

9 Conclusions

In this paper we introduced OMNI, a modelling framework for MAS. The framework supports the development of both closed systems and open, flexible environments. This approach is rather unique, as most existing frameworks concentrate on a specific type of MAS. In OMNI one can specify the organizational structure, the interactions between agents and the normative structure separate from the agents that will populate the MAS. These aspects will ensure that the MAS as a whole, although consisting of autonomous agents, will keep within certain boundaries that are specified for the system as a whole. The framework allows for a very rigid specification of these structures as is needed for open systems like e-institutions, but also allows minimal specifications to allow for emerging behavior of the MAS. The modular structure of OMNI facilitates the adaptation of the framework to different types of domains. In those domains with none or small normative components, design is guided by the Organizational Dimension, while in highly regulated domains the Normative Dimension is more prominent and therefore guides the design. We have provided an example for both cases in this paper to show the different approaches using the framework.

Although we did not specify the formal logics in this paper we should mention that all dimensions of OMNI have a formal logical semantics (cf. [8], [37], which ensures consistency and possibility of verification of the different aspects of the system.

Currently we are taking the first steps towards implementing tools to use with the framework. We will be using Islander as a basis for the support of the implementation level and build the other levels on top of that.

References

1. P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. TROPOS: An agent-oriented software development methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, to appear.
2. C. Castelfranchi. Commitments: from individual intentions to groups and organizations. In V. Lesser, editor, *Proc. 1st. International Conference on Multi-Agent Systems (ICMAS'95)*, pages 41–48. MIT Press, 2005.

3. C. Castelfranchi, F. Dignum, C. Jonker, and J. Treur. Deliberative Normative Agents: Principles and architecture. In *Proc. of the 6th Int. Workshop on Agent Theories, Architectures, and Languages (ATAL-99)*.
4. J. Castro, M. Kolp, and J. Mylopoulos. Towards requirements-driven information systems engineering: the TROPOS project. *Information Systems*, 27:365–389, 2002.
5. M. Dastani, V. Dignum, and F. Dignum. Role assignment in open agent societies. In *Proc. AAMAS'03*, 2003.
6. D. Dennet. *The Intentional Stance*. MIT Press, Cambridge, MA, 1987.
7. F. Dignum. Autonomous Agents with Norms. *AI and Law*, 7:69–79, 1999.
8. V. Dignum. *A Model for Organizational Interaction: based on Agents, founded in Logic*. SIKS Dissertation Series 2004-1. SIKS, 2004. PhD Thesis.
9. V. Dignum and F. Dignum. Modeling agent societies: Coordination frameworks and institutions. In P. Brazdil and A. Jorge, editors, *Progress in Artificial Intelligence*, LNAI 2258, pages 191–204. Springer-Verlag, 2001.
10. V. Dignum, J.-J. Meyer, F. Dignum, and H. Weigand. Formal specification of interaction in agent societies. In *Formal Approaches to Agent-Based Systems (FAABS'02*.
11. V. Dignum, J.-J.CH. Meyer, H. Wiegand, and F. Dignum. An organisational-oriented model for agent societies. In G. Lindemann, D. Moldt, M. Paolucci, and B. Yu, editors, *Proc. RASTA workshop at AAMAS'02*, pages 31–50.
12. V. Dignum and H. Weigand. Towards an organization-oriented design methodology for agent societies. In V. Plekhanova, editor, *Intelligent Agent Software Engineering*, pages 191–212. Idea Group Publishing, 2002.
13. M. Esteva, J. Padget, and C. Sierra. Formalizing a language for institutions and norms. In J.-J.CH. Meyer and M. Tambe, editors, *Intelligent Agents VIII*, volume 2333 of *LNAI*, pages 348–366. Springer Verlag, 2001.
14. D. Grossi and F. Dignum. From Abstract to Concrete Norms in Agent Institutions. In *Workshop on Formal Approaches to Agent-based Systems, FAABS III*, 2004.
15. Nick Howden, Ralph Rnnquist, Andrew Hodgson, and Andrew Lucas. Jack - summary of an agent infrastructure. In *5th International Conference on Autonomous Agents*. 2001.
16. M. Huhns and M. Abdulla. Benevolent agents. *IEEE Internet Computing*, 3(2):96–98, March-April 1999.
17. S. Kalenka. *Modelling Social Interaction Attitudes in Multi-Agent Systems*. PhD thesis, Department of Electronic Engineering, University of London, 2001.
18. R. Likert. *New Patterns of Management*. McGraw-Hill Book Company, New York, 1961.
19. J.-J. Ch. Meyer. A different approach to deontic logic: Deontic logic viewed as a variant of dynamic logic. *Notre Dame J. of Formal Logic*, 29(1):109–136, 1988.
20. J.-J. Ch. Meyer and R.J. Wieringa. *Deontic Logic in Computer Science: Normative System Specification*. John Wiley and sons, 1991.
21. M. Miceli and C. Castelfranchi. The role of evaluation in cognition and social interaction. In Dautenhahn K., editor, *Human Cognition and Social Agent Technology*. John Benjamins, 1999.
22. M. Miceli, A. Cesta, and P. Rizzo. Distributed artificial intelligence from a socio-cognitive standpoint: Looking at reasons for interaction. *Artificial Intelligence and Society*, (9):287–320, 1996.
23. P. Noriega. *Agent-Mediated Auctions: The Fishmarket Metaphor*. Number 8 in IIIA Monograph Series. Institut d'Investigació en Intel.ligència Artificial (IIIA), 1997. PhD Thesis.
24. A. Omicini. Soda: Societies and infrastructures in the analysis and design of agent-based systems. In P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering*, volume 1957 of *LNAI*, pages 185–193. Springer Verlag, 2001.
25. Organización Nacional de Transplantes. http://www.msc.es/ont.
26. H.V.D. Parunak and J. Odell. Representing social structures in uml. In M.Wooldridge, G.Weiss, and P. Ciancarini, editors, *Agent-Oriented Software Engineering II*, LNCS 2222. Springer-Verlag, 2002.
27. W. Powell. Neither market nor hierarchy: Network forms of organisation. *Research in Organisational Behavior*, 12:295–336, 1990.
28. J.A. Rodriguez. *On the Design and Construction of Agent-mediated Electronic Institutions*. PhD thesis, Institut d'Investigació en Intel.ligència Artificial (IIIA), 2001.
29. J. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1969.
30. J. Serrano and S. Ossowski. An approach to agent communication based on organizational roles. In *Cooperative Information Agents VI*, LNAI.
31. J. S. Sichman and R. Conte. On personal and role mental attitudes: A preliminary dependency-based analysis. In F. Oliveira, editor, *Advances in AI: Proc. of the 14th Brazilian Simposium on AI*, LNAI 1515. Springer-Verlag, 1998.

32. M. Singh. Agent communication languages: Rethinking the principles. *IEEE Computer*, pages 40–47, 1998.
33. I. Smith, P. Cohen, J. Bradshaw, M. Greaves, and H. Holmback. Designing conversation policies using joint intention theory. In *Proc. ICMAS-98*, pages 269–276. IEEE Press, 1998.
34. J. van Diggelen, R.J. Beun, F. Dignum, R.M. van Eijk, and J.-J. Meyer. Optimal communication vocubularies in the presence of heterogeneous ontologies. Technical Report UU-CS-2004-003, Institute of Information and Computing Sciences, 2004.
35. W. Vasconcelos, J. Sabater, C. Sierra, and J. Querol. Skeleton-based agent development for electronic institutions. In *Proc. AAMAS'02*, 2003.
36. J. Vázquez-Salceda, U. Cortés, and J. Padget. Integrating the organ and tissue allocation processes through an agent-mediated electronic institution. *LNAI-2504*, pages 309–321, 2002.
37. J. Vázquez-Salceda and F. Dignum. Modelling electronic organizations. In V. Marik, J. Muller, and M. Pechoucek, editors, *Multi-Agent Systems and Applications III*, LNAI 2691, pages 584–593. Springer-Verlag, 2003.
38. G.H. von Wright. On the logic of norms and actions. *New Studies in Deontic Logic*, pages 3–35, 1981.
39. O. Williamson. *Markets and hierarchies: Analysis and Antitrust Implications*. Free Press, New York, 1975.
40. M. Wooldridge, N.R. Jennings, and D. Kinny. The Gaia Methodology for Agent-Oriented Analysis and Design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.
41. F. Zambonelli. Abstractions and infrastructures for the design and development of mobile agent organizations. In M.Wooldridge, G.Weiss, and P. Ciancarini, editors, *Agent-Oriented Software Engineering II*, LNCS 2222, pages 245–262. Springer-Verlag, 2002.
42. F. Zambonelli, N. Jennings, and M. Wooldridge. Organisational abstractions for the analysis and design of multi-agent systems. In P. Ciancarini and M.Wooldridge, editors, *Agent-Oriented Software Engineering*, LNCS 1957, pages 98–114. Springer-Verlag, 2001.