2
Background and Related Research

Research is an activity of continuous work. Expressions such as: 'absolute certainties', 'final answers', 'without any doubt', etc. seem to be expressions too strong to apply to this activity. Moreover, no research work ever stands on its own. Research is for a large extent the recognition of the validity of previous ideas and its applicability to different areas.

This chapter gives an overview of the foundations upon which the research of this thesis is built. Basic theories and related research fields are analyzed in order to provide readers with a better understanding about the concepts and ideas described in the next chapters. Due to the multidisciplinary character of this research, some background on the different disciplines is provided, which may be an overkill for practitioners in that field, but which is aimed to help researchers from the other fields to achieve a common understanding of the work in the remainder of this thesis.

The chapter starts by presenting some work on engineering of MAS, including its phases of modeling and implementation. Then, the field of norm enforcement is investigated. Related work found in each research area is also mentioned here, but each of them will be better exemplified in chapter 6 of this thesis, where they will be compared to DynaCROM.

2.1.
Engineering of Multiagent Systems

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

In the following subsections, some existing solutions for the modeling of MAS as well as others for MAS implementation will be outlined.

2.1.1. Modeling of Multiagent Systems

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

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

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

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

Even with this research effort, organizational approaches have not been a common use in MAS, which is usually seen as a pure aggregation of agents. The fact that organizational approaches have not been effectively adopted suggests that some work still needs to be done in providing better tools for the design and implementation of MAS in which intrinsic characteristics of the application domain (*e.g.*, society structure) can be considered. Moreover, this necessity increases when considering open systems from particular 'cultures'[2].

In the next subsections, two major research lines for the modeling of MAS are presented and discussed. The first research line proposes the modeling of MAS based on organizations and the second one proposes the modeling based on the *electronic institutional* aspects of organizations. By '*electronic institutional*' aspects, the authors mean an organization restricted through the definitions of all the following: related roles, common language, valid interactions and set of norms.

2.1.1.1. Electronic Agent-Based Organizations

The definition of the *organization* term usually varies between two meanings for MAS researchers. In the first meaning, an organization is often understood as an entity with identity that represents (not identical) groups of agents. In the second meaning, an organization is often understood as constraints (structures, norms and patterns) found in a social context that shapes the actions and interactions of agents [Coutinho *et al.*, 2005].

The first sense of an organization comes from an administrative/economic point of view: organizations are like *enterprises* that perform some service or produce some goods. The second sense comes from a sociological point of view: an organization is better called the *social organization* implicitly or explicitly present in a society, community or groups of agents that shape the interactions among agents. These two meanings are not mutually exclusive; the second meaning is more general than the first one. Thus, it is natural to say that every organization (first meaning) has a social organization (second meaning), but the opposite is not always true – every social organization (second meaning) does not always give rise to an organization (first meaning).

---

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

Considering the case that every organization has a social organization, the latter is materialized in the first one by the specification of the structure and objectives of the system. Thus, a social organization is envisioned by the organization as a whole and by describing the activity of the system as realized by the individual agents [Vázquez-Salceda *et al.*, 2005]. In this sense, the organizational dimension covers both the organization and the agent perspectives in the design of agent societies.

The work on MAS modeling based on the organizational dimension mainly started with the emergence of the HarmonIA [Vázquez-Salceda and Dignum, 2003] and OperA [Dignum, 2004] formal frameworks. HarmonIA provides the way to model especially highly regulated electronic organizations from the abstract level, where norms are usually defined, to the final protocols and procedures that implement those norms. The HarmonIA framework also incorporates ontologies to describe and connect different levels of norms.

OperA is a formal specification framework that focuses on the organizational dimension, properly modeling not only organizational structures in an agent society (that structures the global behavior of the society), but also the aims and behavior of the agents from the agent perspective. The framework also explicitly provides a solution for ontological descriptions of agent interactions.

In [Vázquez-Salceda *et al.*, 2005], the *Organizational Model for Normative Institutions* (OMNI) framework is presented, resulting from the combination of some aspects of the HarmonIA and OperA frameworks. The OMNI framework focuses on the organization dimension (that also structures the global behavior of the society), on the behavior of the agents from the agent perspective, on agent interactions and on a normative structure that is separate from the agents that will populate the MAS.

In order to support the development of closed systems and open, flexible environments, OMNI presents a rigid specification of its structure, defining particular fields for the description of scenes, roles and groups of roles. There are no normative aspects further than the ones for organizations, roles, group of roles, agent interactions and agents (only norms for roles, group of roles, scene and transition can be specified). The organization entity is not explicitly present. An organization is formed by listing all its institutional roles (*e.g.*, managers, directors, president, etc.) and represented when agents play these roles. Currently, OMNI does not provide a solution for the implementation and integration of its specifications in a given MAS.

As a possible solution for the limitations of OMNI outlined above, DynaCROM can be used. This way, for instance, OMNI specifications can be transparently applied in a given MAS when agents incorporate the DynaCROM behavior. More details about how DynaCROM can implement OMNI specifications is exemplified in chapter 6 (subsection 6.1) of this thesis, where both solutions are compared.

Another important line of research, based on organizational models for MAS, is mainly proposed by Sichman, Boissier and their colleagues with their work started with $\mathcal{M}$OSE [Hannoun *et al.*, 2000]. $\mathcal{M}$OSE is an organizational model for MAS based on three major concepts: the roles which constrain the individual behaviors of agents, the organizational links that regulate social exchanges between agents and the groups which constrain the layout of agents involved in strong interactions.

In [Hübner *et al.*, 2002], the work on $\mathcal{M}$OSE evolved resulting in the $\mathcal{M}$OISE$^+$ model. $\mathcal{M}$OISE$^+$ permits the specification of a MAS organization along the structural and functional dimensions, which can be specified independently of one another. Furthermore, $\mathcal{M}$OISE$^+$ makes explicit the deontic relation which exists between both dimensions. In short, the $\mathcal{M}$OISE$^+$ organizational model enables the declaration of: the MAS organizational structure (roles, groups and links), functioning (a set of global goals and plans), obligations and permissions.

Comparing $\mathcal{M}$OISE$^+$ with $\mathcal{M}$OISE, the contributions in the structural dimension aim to facilitate the specification (*i.e.*, with the inclusion of an inheritance relation for roles) and to verify if the structure is well formed (*i.e.*, with the inclusion of the compatibility among roles and of a cardinality for roles and groups). Regarding the functional dimension, the main contributions are: the changes in the specification of a mission[3] in order to express the relation among goals and their distribution through the inclusion of a social scheme in the model; the inclusion of the preference among missions; and the inclusion of time in the deontic relations. The functional specification of a mission is represented at a high abstraction level. Nevertheless, this specification could be refined in a more detailed functional description already developed in the MAS area.

[3] A mission is a set of coherent goals that an agent can commit to, according to its roles [Hübner et al., 2002].

The structural, functional and deontic dimensions of the $\mathcal{M}$OISE$^+$ model will be exemplified in chapter 6 (subsection 6.3) of this thesis, where $\mathcal{M}$OISE$^+$ is compared to DynaCROM.

At last, in [Ferber *et al.*, 2009], an integral approach of organizations in MAS, named the MASQ approach, is presented. MASQ integrates the environmental, social and cultural perspectives of a MAS in a four-quadrant framework, where the analysis and design of a system is performed along two axes: an interior/exterior dimension and an individual/collective dimension.

In the approach, an agent may be thought as having two parts: a *mind* and a *body*. The mind corresponds to the agent's decision-making component; and, the ("physical") body determines the agent's existence in the environment, giving to him the ability to act and perceive the environment and its components (by its "social body", *i.e.*, by the role it plays in an organization).

The main asset of MASQ is that, in the approach, an agent's mind may be simultaneously "incarnated" in several bodies from different environments (*i.e.*, *brute spaces*). This capability permits agents to be embedded in one or more cultures, at the same time, and, therefore, to interact with agents from different environments. Over time, an agent can acquire new bodies or leave the bodies it currently possesses. Nevertheless, an agent's body is associated to only a unique mind.

Nowadays, MASQ can be used as an operational framework. However, a methodology based on its principles (mind-body distinction, influence-reaction and organizations centered MAS) still need to be proposed in order to help designers in their decisions on how to model a MAS in terms of culture and brute spaces. Thus, supported by its proposed methodology, MASQ could be used in practice by the MAS designers.

2.1.1.2. Electronic Agent-Based Institutions

The idea of modeling MAS as institutions came from the observation that human institutions [North, 1990] have been successfully mediating human interactions for centuries and, so, EI (meaning *Electronic Institution(s)*) may cope with a similar responsibility within agent societies. The aim of the proposal is to promote a natural extension of human institutions by permitting not only humans, but also autonomous agents to interact with one another in a reliable way. Thus,

in short, an EI can be seen as the electronic counterpart of a human institution in which interactions between agents are articulated through a role-based multiagent protocol specification.

The work on formalization of EI has been done for years and it is extensively presented mainly in [Noriega, 1997; Rodríguez-Aguilar, 2001; and, Esteva, 2003]. In [Noriega, 1997], the different components of an institution are introduced by using a typical trading institution – the fish market auction houses – as a motivating example. Noriega proposes that an institution is defined: (i) by a set of roles and relationships within them, (ii) by a common ontology and communication language which allow heterogeneous agents to exchange knowledge, (iii) by the valid interactions that agents may have structured in conversations, and (iv) by a set of rules of behavior which determine the actions that agents must take under certain circumstances.

In [Rodríguez-Aguilar, 2001], the formalization of EI presented by Noriega was extended and refined, resulting in the definition of ways of realizing EI. Rodríguez-Aguilar proposes an infrastructure to implement EI that can be realized by making use of a special type of mediator agents, the so called *interagents* [Martín *et al.*, 2000].

Each agent involved in a conversation is connected to an interagent, which mediates the agent's interactions in one-to-one conversations. Conversations among many agents (more than two, at least) cannot be managed by an interagent in the infrastructure proposed. A main feature of interagents is that the conversation protocol that they manage is from the agent's point of view and, so, it is not an overview of protocol as a whole.

As a proof-of-concept for his infrastructure, Rodríguez-Aguilar presented a detailed description of the fish market implementation.

In [Esteva, 2003], the previous work done by [Noriega, 1997; and, Rodríguez-Aguilar, 2001] on the formalization of EI was continued. In his work, Esteva provides support for the specifications of EI, their automatic verification and also their realization. His main concrete result, the ISLANDER graphical editor, was developed as a generic infrastructure which could be used for the deployment and verification of the specified institutions.

The limitation of the Rodríguez-Aguilar's work in which only one-to-one conversations could be mediated by interagents was improved in Esteva's work. There, for each conversation, a governor agent (an evolution of the interagent one) has two queues, one for the messages received from its associated agent and another one for the messages received from the social layer agents. As a

case study, Esteva evolved the previous examples of Noriega and Rodríguez-Aguilar on fish markets, now regarding multi-market institutions instead of only single-market ones.

Many other publications of EI have appeared recently [*e.g.*, Esteva *et al.*, 2004; García-Camino *et al.*, 2005 and 2006; Grossi *et al.*, 2007], expanding the work on the subject.

In [Esteva *et al.*, 2004], the AMELI agent-based middleware is proposed as an infrastructure that mediates agents' interactions while enforcing institutional norms. The combination of ISLANDER and AMELI supports the design and development of open MAS adopting a social perspective.

In [García-Camino *et al.*, 2005], a distributed architecture for EI is proposed in order to endow MAS with a social layer in which normative positions are explicitly represented and managed via rules for regulation. In [García-Camino *et al.*, 2006], the rule-based language from the authors is better detailed as a declarative normative language that can represent distinct flavors of deontic notions and relationships. Every external agent from the architecture has a dedicated governor agent linked to it that enforces the norms of executed events.

In [Grossi *et al.*, 2007], the work on formalization of EI is continued, focusing on both institution and its components (abstract and concrete norms, empowerment of agents and roles). Yet, a formal relation between institutions and organizational structures is also defined in such a way that institutional norms can be refined to construct organizational structures, which are closer to an implemented system. Thus, the gap between abstract norms and concrete system specifications is better bridged.

Despite all work done, a MAS implemented as an EI is still understood as a type of dialogical system that simply structures agent interactions by establishing the commitments, obligations and rights of participating agents. However, the solution not only structures interactions, but also enforces individual and social behaviors by obliging every agent to act according to the defined norms.

The following current limitations of EI are outlined: (i) there are no normative aspects further than the ones for roles, agent interactions and agents; (ii) the specification of an EI is often too *society-centric* in the sense that it completely fixes agent interactions in rigid protocols and interfaces; (iii) external agents have no room for autonomous behavior, *i.e.*, they blindly follow defined protocols with the only autonomy to accept or reject them; (iv) all possible interactions among agents have to be defined; (v) it is difficult, if not impossible, to describe indirect interactions; this is due to the fact that all interacting activity taking place

in an EI is purely dialogic by means of direct communication between the agents; and, (vi) the structure of an EI is static and, so, cannot evolve at system runtime.

These limitations will be further treated in chapter 6 (subsection 6.2) where DynaCROM is compared with EI solutions (ISLANDER and AMELI).

2.1.1.3. Discussion

The models used to describe or design an organization are classically divided into the *agent-centered* or *organizational-centered* perspectives [Lemaître and Excelente, 1998]. In the first perspective, system developers try to analyze and/or design a whole MAS that shows a non-accidental and non-chaotic global behavior starting from the agents (parts of the system).

In the open MAS scenario, the basic problem with the *agent-centered* idea is that the system developer has no control anymore over the creation of the agents. Thus, at any time, external heterogeneous agents can join or leave an open MAS and, then, disrupt the existing order. As long as open MAS are highly desirable to face today's increasingly distributed and interconnected computing demands, this wish poses problems that still need concrete solutions.

In the last few years, one promising path of research and development has been an *organizational-centered* analysis and design of MAS (second perspective). In this attempt, system developers proceed in a top-down fashion, explicitly defining both the organization entity (external to the agent level) and the organization statutes that agents must comply with. The statutes of an organization indicate, at the most abstract level, the main objectives of the organization and the values that direct the fulfilling of its objectives. Moreover, statutes also point to the context in which the organization will have to perform its activities [Vázquez-Salceda *et al.*, 2005].

Analyzing several *organizational-centered* models found in the literature (*e.g.*, OMNI [Vázquez-Salceda *et al.*, 2005], ISLANDER [Esteva *et al.*, 2002], $\mathcal{M}$OISE$^+$ [Hübner *et al.*, 2002]), we agree with [Coutinho *et al.*, 2008a] about the two main sources of difficulties found on *organizational-centered* models.

The first source of difficult is that the very notion of organization admits and is frequently used with slightly different interpretations. Sometimes, the organization term refers to "*collectivities oriented to the pursuit of relatively specific goals*

and exhibiting relatively highly formalized social structures" [Scott, 1998]. Other times, the term refers to stable social patterns/structures of joint activity that constrains and drives the actions and interactions of agents towards a purpose.

The second source of difficulty is that the organization entity can be described in several modeling dimensions (*e.g.*, in the structural and functional ones).

These two sources of difficulties of *organizational-centered* models are important and should be considered because each proposal of an organizational model makes a particular ontological commitment in regard to them.

A proposal for an integrated ontology, which is developed in a bottom-up manner from the existing organizational models, is presented in [Coutinho *et al.*, 2008a]. The main purpose of such ontology is the creation of an interoperation mechanism that can be used by heterogeneous organizational models for handling interoperability among open *organizational-centered* MAS. However, the proposal is an ongoing work and, therefore, needs to be concluded.

In [Vázquez-Salceda *et al.*, 2005], the following drawbacks of current approaches for MAS modeling also are pointed out:

- MAS modeling are too *agent-centric* or too *organizational-centric*. Some methodologies (*e.g.*, GAIA [Wooldridge, 2000]; Prometheus [Winikoff and Padgham, 2004]) are too *agent-centric*, in the sense that they 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. Other methodologies (*e.g.*, SODA [Omicini, 2001] and ISLANDER) are too *society-centric* in the sense that they completely fix agent interactions in rigid protocols and interfaces. Thus, agents cannot exercise their characteristic of autonomy.
- *Roles* and *agents* are usually treated without an explicit distinction. This distinction is an important asset in order to establish a difference between organizational values and individual (agent) values.
- Normative aspects are not often considered or, when considered, they 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 (by means of the usage of policies or protocols).

- Ontologies are seen as an external (accessory) component, while in fact they are tightly coupled with the rest of the system when used to model most of its elements.

2.1.1.4. Comparative Study

Table 1[4] summarizes a comparative study conducted among the main presented modeling solutions for the engineering of MAS. In the study, the following research questions are proposed for each solution analyzed:

- rq.i. Does it explicitly support the organizational normative dimension?
- rq.ii. Does its conceptual model have an implemented solution for it?
- rq.iii. Does it support the management of norms to be done at system runtime?
- rq.iv. Does it provide ways for norm representation with a common understanding for heterogeneous agents?
- rq.v. Does it have an editor, preferably a graphical one, to support the writing of its specifications?
- rq.vi. Does it have a semi-/automatic solution for the verification of its specifications?

Table 1. A comparative study conducted among modeling solutions for MAS engineering

	rq.i	rq.ii	rq.iii	rq.iv	rq.v	rq.vi
OMNI	$\mathcal{V}$	X	–	$\mathcal{V}$	X	X
$\mathcal{M}$OISE$^+$	$\mathcal{V}$	$\mathcal{V}$	$\mathcal{V}$	$\mathcal{V}$	X	X
ISLANDER	X	$\mathcal{V}$	$\mathcal{V}$	$\mathcal{V}$	$\mathcal{V}$	$\mathcal{V}$

2.1.2. Implementation of Multiagent Systems

In order to implement complex systems, with a high degree of interoperability, system developers should follow the specifications defined by FIPA (mean-

[4] In the table presented above and in following ones, '$\mathcal{V}$' means 'Yes', 'X' means 'No' and '–' means 'Not Applicable'.

ing *Foundation for Intelligent Physical Agents*) [FIPA, URL]. FIPA is an IEEE Computer Society standards organization for agents and MAS. Its aim is to promote agent-based technology and interoperability of its standards with other technologies.

In the following subsections, JADE and ASF, two agent platforms for implementing MAS compliant with the FIPA specifications, are presented.

2.1.2.1. The JADE Agent Platform in Brief

In this section, an overview of JADE (meaning *JAVA Agent Development Environment*) is given. For a more detailed description of JADE, readers are referred to [JADE, URL; and, Bellifemine *et al.*, 2001, 2002a and 2002b].

JADE is a software development framework which contains a FIPA compliant agent platform, developed in JAVA [JAVA, URL], and a package to develop JAVA agents. JADE also offers facilities for the definition of user ontologies and new content languages. The main goal of JADE is to simplify and facilitate the development of MAS. The platform is perceived from outside as a single entity, but it can be divided into different agent containers, which can be distributed in different hosts.

Each agent executes in JADE within an agent container that, in turns, executes within a JAVA virtual machine. In the platform, each agent must have a global unique identifier which is composed by the names of both the agent and the platform where it is running. Concretely, the global unique identifier of an agent is constructed, like e-mail addresses, by the composition of the name of the agent, a '@' symbol and the address of the platform. Agents can freely migrate among different containers, from their same hosts or from different ones.

JADE defines a generic agent class that agent developers must extend in order to program their agents. Developed agents inherit, from the JADE agent super-class, different services, such as: the capability of registering and deregistering in the platform, a set of basic methods for sending and receiving messages, the capability of cloning, etc.

In JADE, agent functionality is defined by a set of behaviors, which can be either added or removed dynamically. Behaviors are represented by logical threads, but each agent has only one associated thread being executed at a given moment. During its execution, each agent maintains a queue of active behaviors, a queue of blocked behaviors and a queue of incoming messages. When

a behavior finishes, the next one in the queue of active behaviors is executed. When an agent receives a new message, this message is added to its queue of incoming messages and, then, all blocked behaviors related to the message are activated by moving those behaviors to the queue of active behaviors. Messages in the queue can be accessed by blocking and unblocking operations and, in both cases, a message template can be verified. In the case of verification, the returned message must match the given template. For blocking operations the maximum time to wait for messages can be defined.

Agents within JADE exchange messages in FIPA ACL. Agent containers communicate using JAVA RMI (meaning *JAVA Remote Method Invocation*) [JAVA RMI, URL] while, for communication with other FIPA compliant platforms, IIOP (meaning *Internet Inter-Orb Protocol*) [IIOP, URL] is used.

JAVA RMI enables programmers to create distributed JAVA technology to JAVA technology-based applications, in which the methods of remote JAVA objects can be invoked from other JAVA virtual machines, possibly on different hosts. RMI uses object serialization to assemble and disassemble parameters, and does not truncate types, supporting object-oriented polymorphism.

IIOP is an implemented protocol that rides on the top of transport protocols, such as TCP/IP[5], delivering distributed computing capabilities of CORBA (meaning *Common Object Request Broker Architecture*) [CORBA, URL]. For instance, a client can transparently invoke a method on a server, which can be on the same machine or across a network. Another example, browsers and servers can exchange more complex objects (*e.g.*, integers, arrays) instead of merely transmitting texts.

2.1.2.2. The Agent Society Framework in Brief

Besides agents, several authors identify other essential concepts that should be considered when implementing a MAS. In [Jennings, 2000], the author identifies *agents*, *organizations*, *interactions* and *environments* as central MAS concepts. In [Ferber *et al.*, 2000], the authors define *agents*, *roles* and *groups* (or *organizations*) as the main concepts of a multiagent society. [Dastani *et al.*, 2003] and [Depke *et al.*, 2001] agree that the position of an agent in its organiza-

[5] TCP/IP (meaning *Transmission Control Protocol/Internet Protocol*) is a suite of communications protocols used to connect hosts on the Internet.

tion is characterized by assigned roles and, moreover, that agents inhabit environments. In the context of organizations, the authors of [Zambonelli *et al.*, 2001] suggest that the autonomous behavior of agents should be designed in the models of both behavior and structure of human organizations.

Regarding these references, an agent architecture, created to design and construct MAS in which agent societies are present, should define agents and their interactions, organizations, roles and environments as first class abstractions.

ASF (meaning *Agent Society Framework*) [Silva *et al.*, 2004a] is an object-oriented framework implemented for supporting the design and construction of agent societies. The main asset of ASF compared to other agent-based frameworks/platforms (*e.g.*, JADE, FIPA-OS [Poslad *et al.*, 2000], Zeus [Nwana *et al.*, 1999], Kaos [Bradshaw *et al.*, 1997]) is that it defines an agent application architecture that supports all the essential concepts of MAS (*i.e.*, agents, interactions, organizations, roles and environments).

ASF is composed of sets of related object-oriented modules, in which each module represents an agent entity, and it is implemented by a set of classes and relationships. In each set of classes, an agent entity is represented by an abstract class and agent properties (*e.g.*, pro-activity, adaptation, mobility) are represented by concrete or abstract classes. This way, the main entities of an agent society are implemented by using an object-oriented framework in which each set of classes embodies an abstract design for each entity of the society.

2.1.2.3. Discussion

Because JADE is probably the widely used platform for MAS implementation in academic circles and also because it is in conformance with the FIPA specifications, then, JADE was chosen to be used in the two usage scenarios developed for this thesis (both presented in chapter 5). There, the JADE limitation of only providing containers for holding the agents that participate in the implemented MAS resulted in a mix of domain concepts (*e.g.*, *environment*, *organization*, *political*, etc.) in the same level of abstraction.

That limitation is minimized in ASF because the framework considers agents, organizations, roles and environments as first class abstractions in MAS. However, ASF, as JADE, does not have an explicit support for specific domain concepts (*e.g.*, *political* and *economical* concepts) and, yet, ASF is not as popu-

lar as JADE. Therefore, the effort of implementing in ASF is limited to a proof-of-concept presented in [Felicíssimo *et al.*, 2005a]. There, the implementation of some essential concepts for a MAS from the traffic urban domain (*e.g.*, traffic signs, urban paths and places, drivers, pedestrians, etc.) is described.

2.1.2.4. Comparative Study

Table 2 summarizes a comparative study conducted between the presented implementation solutions for the engineering of MAS. The following research questions outline the reasons for the choice of the platform used in the two usage scenarios of this thesis:

- rq.i. Does it permit the mobility property of agents?
- rq.ii. Does it support the direct implementation of agent societies?
- rq.iii. Is it a well-known platform for MAS implementation?

Table 2. A comparative study conducted between implementation solutions for MAS engineering

	rq.i	rq.ii	rq.iii
JADE	$\surd$	X	$\surd$
ASF	X	$\surd$	X

2.2. Norm Enforcement

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

Regarding these distinctions, an appropriate solution for regulating interactions among agents cannot be rigidly fixed at any system phase and should continuously support data updates according to the changing contexts of agents.

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

Regarding '*situation of participants*', this work is concerned with the issue of regulation in complex systems, which follows [Simon, 1996] in his definition stating that: "*complexity frequently takes the form of a hierarchy. That is, a system is composed of interrelated subsystems, each of which is in turn hierarchic in structure, until the lowest level of elementary subsystem is reached*". Moreover, besides hierarchical relationships among participants in interrelated subsystems, non-hierarchical relationships are also considered in this thesis due to the reason that information from those relationships can be relevant.

Thus, it makes it necessary to provide a contextual normative solution in which different types of relationships among agents can be dealt with in order to enable norm enforcement in NMAS. The solution should be flexible enough for supporting norm evolution and it should not be only based on the interaction level, but also on others domain levels.

In the following subsections, some current solutions found in the area of norm enforcement will be presented. The idea is to give an overview about such solutions instead of getting into their operational details. Those details are further explored in chapter 4 of this thesis. There, the way DynaCROM outputs (*i.e.*, agents' contextualized norms) are used as a precise input for norm enforcement solutions will be described.

2.2.1. The Moses Mechanism for Implementing LGI Specifications

The efforts of using laws as a regulatory solution emerged when Minsky published the [Minsky and Rozenshtein, 1987] work with his seminal ideas about the creation of a law-governed object-oriented system. In the work, the concept of agents is restricted to the active part of an object that sends and receives (controlled) messages.

In [Minsky and Ungureanu, 2000], Minsky's work evolved to the implementation of interaction norms in distributed systems by using LGI, which is proposed as a decentralized coordination and control mechanism. The concept of agents is explicit and MAS are defined as conglomerates of semi-autonomous, heterogeneous and independently designed subsystems, constructed and ma-

naged by different organizations with little, if any, knowledge of each other. LGI is implemented by the Moses toolkit (also presented in the paper) and it is based on four principles. These principles were proposed for the coordination within open groups of autonomous and distributed agents, and for the control of such groups, as follows:

- Principle 1: ENFORCEMENT. "*A coordination policy for an open group needs to be enforced*".
- Principle 2: DECENTRALIZATION. "*The enforcement mechanism should not require central control*".
- Principle 3: SEPARATION OF POLICY FROM MECHANISM. "*Coordination policies should be made explicit and be enforced by means of a single mechanism that can implement a wide range of policies in a uniform manner*".
- Principle 4: INCREMENTAL DEPLOYMENT. "*One should be able to deploy and enforce a policy incrementally, without exacting any cost from agents and activities not subject to it*".

Since then, the work on design in LGI and on implementation in Moses, both for coordination and control in large, open and distributed systems, continuously evolved in a wide range of domains, including: e-commerce, grid computing, enterprise systems, self-healing and MAS, as presented in several publications found in [Minsky_Publications, URL]. Although the Minsky's publications are important for the chapter, its extensive list will not be explained. However, the following three publications were arbitrary chosen to be presented in order to illustrate current evolutions in LGI and, consequently, in Moses.

In [Ao *et al.*, 2000], the notion of *communities* in LGI is introduced. *Communities* are represented by implicit groups (such as the ones found in the Internet), which require no central control of any kind and whose membership does not have to be regulated – it might be completely unknown to anyone. This notion is opposite to the original one of explicit groups in LGI, which is necessary for applications that require each member of a group to know about the membership of the entire group. The details about how implicit groups were implemented to enable agents operating on it to interact and how MOSES was extended to provide the implementation are also presented in the paper.

In [Ungureanu and Minsky, 2000], it is presented another extension of LGI that provides interoperation between heterogeneous policies, formulated inde-

pendently by different authorities, allowing inter-enterprise electronic commerce. The four LGI principles were then refined for application domains, as follows:

- Principle 1: "*E-commerce policies should be made explicit and enforced by means of a generic mechanism that can implement a wide range of policies in a uniform manner*".
- Principle 2: "*The enforcement mechanism of e-commerce policies needs to be decentralized*".
- Principle 3: "*Inter-operation between e-commerce policies should maintain their privacy, autonomy and mutual transparency*".

In [Serban *et al.*, 2001], the main ideas of the two previous works (communities and interoperability among heterogeneous policies of enterprises) were put together. The result of the work is a solution for the implementation of policies that are supposed to govern the various communities operating within an enterprise.

2.2.1.1. Discussion

In short, Moses is a decentralized coordination and control mechanism for distributed systems that implements LGI. LGI enables a distributed group of actors – which may be heterogeneous, open and large – to engage in a mode of interaction governed by an explicitly specified policy called the *law* of this group.

Although Moses has been used in a variety of application domains as a well-known solution for law enforcement in distributed systems, Moses still has some limitations while enforcing norms in a MAS. The first limitation of Moses is that it does not offer the support to directly enforce norms that are not of interaction (*e.g.*, *environment* and *organization* norms). Thus, it makes it necessary to decouple norm information from different levels of abstractions to the interaction one.

The second limitation of Moses is that an agent is restricted to interact only with the other agents of its own community. A Moses community is formed by (Moses) agents operating under a unique law. Only agents playing in the same community (*i.e.*, operating under the same law) can interact. If agents from distinct communities decide to interact, it will not be possible because they are operating under different laws.

For instance, Figure 2 part 1 illustrates the ‘agent x’ interacting with an ‘agent y’, both operating under a law called “simple.java1”. The ‘agent x’ cannot operate with any other agent not operating under the “simple.java1” law, as, for example, with the ‘agent z’, which is operating under a law called “ping.java1” (the hashed line in the community represents that the interaction between the ‘agent x’ and the ‘agent z’ is not possible).

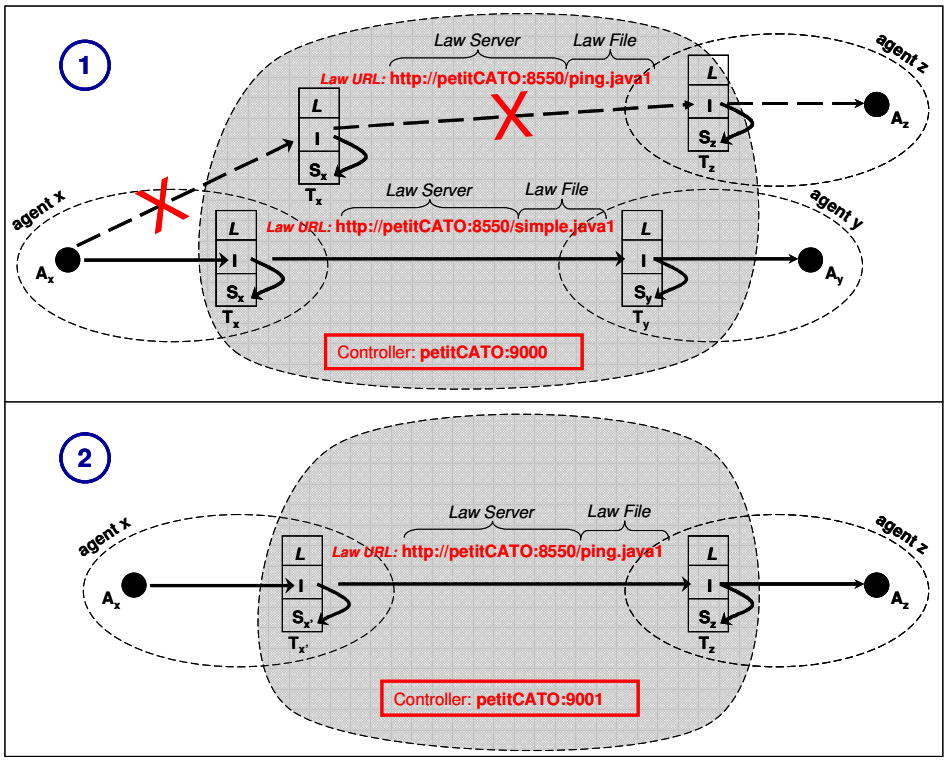

Figure 2 – Agents from a same community operating under a unique law

A possible solution for agents from different communities to interact is by creating agents’ copies for each community (represented by different Moses controllers). An agent and its copies have the same original capabilities (*e.g.*, the expertise to play a ping-pong game). However, during the agents’ life cycle, an agent and its copies are independent entities (possibly, with new capabilities acquired).

Figure 2 part 2 illustrates a copy of the ‘agent x’ interacting with the ‘agent z’, both operating under the “ping.java1” law. Agent copies are created in different Moses controllers (*e.g.*, “Controller1: petitCATO: 9000” and “Controller2: petitCATO:9001”) for permitting agents to maintain their (unique) original names. Thus, agent copies and their origins are differentiated by the controllers where they play and by the laws that they are subject to.

The third limitation of Moses is that it lacks dynamics while evolving law information in communities. This is because a Moses community operates under a unique *static* law that must be already created when agents join it. If any evolution or even a repair is necessary in a law, its associated community has to be paused for the updates and, then, restarted.

Moses will be detailed in chapter 4 of this thesis, where it is presented as a solution for norm enforcement based on the agents' external behavior. The applicability of the integration of DynaCROM and Moses will be exemplified, in chapter 5, with the aid of two usage scenarios.

2.2.2. The AMELI Middleware for Implementing ISLANDER Specifications

In [Esteva *et al.*, 2004], the AMELI agent-based middleware for EI is presented as an infrastructure that mediates agents' interactions in order to enforce institutional norms. The enforcement is done in terms of: guaranteeing the correct evolution of each scene execution (*i.e.*, by filtering erroneous illocutions for the prevention of errors made by participating agents); guaranteeing that movements of agents between scene executions comply with the specification; and controlling which obligations participating agents acquire and have to fulfill.

These enforcement functionalities are performed by the following four types of agents:

- Institution Manager. An *institution manager* agent is in charge of starting an EI, authorizing agents to enter the institution, as well as managing the creation of new scene executions. The agent keeps information about all participants and all scene executions. There is one institution manager per institution execution.
- Transition Manager. A *transition manager* agent is in charge of managing a transition that controls agents' movements in several scenes. There is one transition manager per transition.
- Scene manager. A *scene manager* agent is responsible for governing a scene execution. There is one scene manager per scene execution.
- Governor. A *governor* agent is devoted to mediate the participation of an external agent within the institution. There is one governor per participating agent.

The middleware is characterized in the AMELI infrastructure by the presence of two layers: a *public layer*, in which external agents interact with governor agents, and a *private layer*, in which internal agents interact with their governors and is inaccessible to external agents.

Each governor agent mediates all interactions of the participating agent to each one it is connected to. For each message received, the governor replies to its participating agent with one of the following messages: *agree* for correct messages, *refuse* for incorrect messages, or *unknown* for messages not understood. Correct messages are processed later on, considering the context of the conversation it belongs to.

Governor agents also have the responsibility to manage norms by using a rule-based system, which has two types of rules: one rule for norm activation and another one for fulfillment of obligations. The first type of rules needs to be added by governors, however, the second one will be added and removed dynamically in governors' rule bases as obligations are acquired or fulfilled.

2.2.3. The *S*-*M*OISE$^+$ Middleware for Implementing *M*OISE$^+$ Specifications

S-*M*OISE$^+$ [Hübner *et al.*, 2006] is a middleware developed to enforce, at agents' runtime, constraints, which are specified by following the *M*OISE$^+$ conceptual model [Hübner *et al.*, 2002].

S-*M*OISE$^+$ has two main components: an API and a special manager agent. The API is used by the agents of the system to access the organizational layer. The special manager agent has the current state of the organizational entity (OE) and the responsibility to maintain this state consistent. The manager agent receives messages from the user agents requesting for changes in the OE state (*e.g.*, role adoption, group creation, mission commitment, etc.). Then, the manager agent changes the OE only if the agents' requests do not violate any organizational (predefined) constraint, by so, maintaining the OE consistent.

The aim of the *S*-*M*OISE$^+$ middleware is a tentative to fill the gap between organizational constraints and agent autonomy. However, because a proper communication layer still needs to be improved, user agents can directly interact

with the manager agent by using KQML [Finin *et al.*, 1993] or FIPA-ACL [FIPA ACL, URL]. In this case, the communication link constraint is not guaranteed, since agents get direct access to the communication layer.

2.2.4. SCAAR

SCAAR [Chopinaud *et al.*, 2006] (meaning *Self-Controlled Autonomous Agents geneRator*) is a norm enforcement mechanism that enhances agents with a self-monitoring capability for avoiding norm violation.

SCAAR automatically adds control hooks and an enforcement core to the agents' codes when agents incorporate the SCAAR solution. Control hooks can be inserted inside agents' code before a regulated action, to prevent norm violation, or after, to detect norm violation. Once a regulated action starts running, its control hook triggers the agent enforcement core for the verification and/or enforcement of norm compliance.

If the system developer decides to use the SCAAR norm prevention mechanism in his NMAS, then, in case of an attempt to violate an obligation or prohibition norm, the enforcement core blocks the execution of the infringing action and informs it to its agent. If the system developer decides to use only the SCAAR norm detection mechanism, then, when a norm violation occurs with an obligation or prohibition norm, the enforcement core informs it to its agent. For a permitted norm, no specific action is taken by SCAAR.

SCAAR will be detailed in chapter 4 of this thesis, where it is presented as a solution for norm enforcement based on the agents' internal behavior. The applicability of the integration of DynaCROM and SCAAR will be exemplified, in chapter 5, with the aid of two usage scenarios.

2.2.5. The M-Law Middleware for Implementing XMLaw Specifications

M-Law [Paes *et al.*, 2006 and 2007a] is a middleware developed to enforce, at agents' runtime, interaction laws, which are specified by following the XMLaw conceptual model. The middleware allows extensibility for open system requirements and interoperability concerns.

XMLaw [Paes *et al.*, 2005] is a conceptual model composed of the following related elements, presented with their definitions:

- *Event*: models the occurrence of a law element;
- *State* (one of three types: *successful*, *failure* or *execution*): represents a static or dynamic situation in the evolution of an agent's interaction;
- *Protocol*: defines the possible *states* that an agent interaction can evolve to;
- *Transition*: represents the change occurred in the course of an interaction caused by the response to the occurrence of an *event*;
- *Scene*: models an interaction context in which protocols, actions, clocks and norms can be composed to represent complex normative situations through a set of behavioral rules or social conventions;
- *Clock*: represents time restrictions or controls and can be used to activate other law elements;
- *Norm* (one of three types: *permission*, *obligation* or *prohibition*): is used to enable or disable agents' conversation paths or law events according to its predefined conditions;
- *Constraint* (implemented by using JAVA code): is defined inside a scene or a norm element as a restriction to norms or transitions that generally specifies the allowed values (*i.e.*, filters) for events.
- *Action* (also implemented by using JAVA code): is used to plug governance services in M-Law in three different scopes: law, scene and norms.

Currently, M-Law has been used to employ Dependability Explicit Computing (DepEx) ideas in the construction of dependable open MAS [Paes *et al.*, 2007b]. The dependability of a system can be defined as the ability to avoid service failures that are more frequent and more severe than is acceptable [Avizienis *et al.*, 2004]. Flexibility is achieved in M-LAW by using an event-driven approach, at a high-level of abstraction, in which law elements communicate by the exchange of events.

2.2.6. Discussion

Although it is possible to specify very complex interaction norms by using abstractions of low level, this is not reasonable when designing complex sys-

tems in which domain norms are also necessary. This is because the task of (several) mappings of high level norms to low level ones is arduous and, moreover, the purpose of domain norms are lost once they are scattered in many low level primitives.

This situation reflects the problem that solutions for norm enforcement are both disconnected from the application domain and restricted to regulation only at the interaction level.

Hence, the development of complex and interactive systems demands high level abstractions for regulation, as the support for implementing domain norms, which have to be manipulated in a simple way, not increasing the difficulty of the development task, but minimizing it.

Moreover, regarding that *"agents from different organizations have little, if any, knowledge of each other*" [Minsky and Ungureanu, 2000] and that "*the less autonomous an agent is, the more interactive it needs to be to achieve its goals*" [Silva, 2004c p. 34], it is necessary to guarantee autonomy for agents in open domains. This will make it possible for agents to execute in high levels without having to interact with others in low levels.

2.2.7. Comparative Study

Table 3 summarizes a comparative study conducted among the presented solutions for norm enforcement. The objective of the study is to outline the reasons why the solutions for norm enforcement were chosen to be used by DynaCROM. In the study, the following research questions are proposed for each solution analyzed:

- rq.i. Does it explicitly support the implementation of an organizational normative dimension?
- rq.ii. Does it have a manager/governor/police agent for norm enforcement?
- rq.iii. Does it directly enforce prohibition norms?
- rq.iv. Does it make a distinction between the implementation of an organization entity and a group of roles? *I.e.*, Does the organization concept have an explicit entity to support its implementation?
- rq.v. Can the structure of the system (in any dimension, *e.g.* the normative one) evolve at MAS execution time?

rq.vi. Does the implementation have a conceptual model to guide its specifications?

Table 3. A comparative study conducted among norm enforcement solutions

	rq.i	rq.ii	rq.iii	rq.iv	rq.v	rq.vi
Moses	X	$\mathcal{V}$	X	X	–	X
AMELI	X	$\mathcal{V}$	X	X	X	$\mathcal{V}$
$\mathcal{S}$-$\mathcal{M}$OISE$^+$	$\mathcal{V}$	$\mathcal{V}$	X	X	$\mathcal{V}$	$\mathcal{V}$
SCAAR	X	X	$\mathcal{V}$	–	–	X
M-Law	X	$\mathcal{V}$	X	–	–	$\mathcal{V}$

2.3. Discussion

In this chapter, the foundations upon which the research of this thesis is built are presented by describing some work on engineering of MAS, including its phases of modeling and implementation, and on norm enforcement.

A comparative study was done for each work on the research lines analyzed and the conclusion of the study is that no current solution answers all the research questions posed in the section 1.2.1 of this thesis. So, those questions guide the work of this thesis to achieve the objectives presented in the section 1.2.2 that, in turn, lead to the milestones of DynaCROM, as will be presented in the next chapter.