A UML-Based Notation for Representing MAS Organizations

Massimo Cossentino*, Carmelo Lodato*, Salvatore Lopes*, Patrizia Ribino*, Valeria Seidita†, Antonio Chella†

*Istituto di Reti e Calcolo ad Alte Prestazioni,
Consiglio Nazionale delle Ricerche,
Palermo, Italy.
Email: {cossentino, ino, toty, ribino}@pa.icar.cnr.it

†Dipartimento di Ingegneria Chimica Gestionale Informatica Meccanica
Università degli Studi di Palermo
Email: {seidita, chella}@dinfo.unipa.it

***Abstract*—A notation for representing agents' organizations to be implemented using Moise+ and Jason is proposed. For this purpose a UML profile was defined for representing the elements of Moise+ organizational model such as role, mission and group. The proposed notation will be fully illustrated and applied to the classical example provided by the J-Moise+ team.**

I. Introduction

In the context of highly complex, distributed and open systems, engaged and working in dynamic environments are widely employed. Such system should include the capability of continuously reacting, with a re-organization process, to changes occurring in the environment. Because of their intrinsic nature, agents have been recognized to be a good way for solving complex problems both at the design and the implementation levels [1] [2].

Organizations [3] play a relevant role in multi-agent systems design; they can be seen as the set of constraints ruling the agent's behavior in multi-agent systems (MAS from now on).

As regards the agent organization implementation, a robust approach coming from Hubner et al. [4] proposes an organizational model (Moise+) able to support the re-organizing process of MAS. Moise+ describes the organization in a MAS by employing three main views: the structural, the functional and normative specifications. In this model an organization is established a priori (created at design-time) and the agents ought to follow it. The Moise+ organizational model considers the structural and functional dimensions as almost independent while the normative dimension is used to establish a link between them.

The Moise+ organizational model is complemented with the possibilities of quickly and easily programming MAS by means of J-Moise+ [5], a Jason extension allowing developers to use Jason for programming agents and their organizations [6].

This offers a powerful tool to MAS developers, nevertheless it is not still adequately supported by a well defined methodological approach. Some researchers in the past developed methodologies for MASs where some aspects of organization were modeled. In one of the most known in literature [7] the concepts of environment, roles, interactions and organizational rules are taken into account as organizational abstractions. Another example has been proposed in [8] where holarchy represents the organization structure of the MAS made of holons [9] hence the main element to be developed for building the MAS organization.

The work illustrated in this paper regards the creation of a specific notation for representing the organizational model proposed by Moise+. The advantages of having a graphical notation for representing organizations are evident: first of all, graphical notations are more readable and understandable at a glance than any coding language, secondly it is usually easier to explain a graphical notation to stakeholders involved in the designer (that are not designer) than read the application code with them. The possibility of involving stakeholders like system users enables the adoption of agile or extreme development approaches and improves the flexibility of conventional ones.

The remainder of the paper is organized as follows. In section II the Moise+ organizational model, J-Moise+ and Jason are introduced. In section III we explain the proposed notation by using three kinds of diagram in order to define graphically the structural, functional and normative specification of a Moise+ organization. In this section an instance of the notation in use by using the Moise+ tutorial [10] example for generating the three specification diagrams is also provided. Section IV offers a comparison with others MAS modeling proposals. Finally some discussions and conclusions are drawn in section V.

II. Background

A. Moise+

Moise+ [11][4] is an organizational model for MAS looking at organization from three different perspectives: structural, functional and normative. From the structural viewpoint, an organization can be seen as a set of *Roles* linked by *Relations*

and clustered into *Groups*. Analyzing an organization from the functional perspective allows designers to define the global objective, and also the plans and the way for reaching this goal by means of a *Social Scheme*. In this scheme the functionalities of the organization are represented as *Goals* grouped into *Missions*. Finally, modeling the normative aspect of the organization allows to assign a mission to a Role by means of *Permission* or *Obligation* norms. Norms can be seen as the backbone connecting the functional and structural aspects of an organization.

While the Moise+ implementation is based on two key elements: the Organizational Specification (OS) that is the union of structural, functional and normative specification and the Organizational Entity (OE) that is the instantiation of OS on a set of agents.

B. Jason

The development of cognitive agents can be based on different approaches [12]. Jason approach is based on the BDI (Belief-Desire-Intentions) architecture characterized by the implementation of agent's beliefs, desires and intentions. The AgentSpeak [13] is an abstract agent language founded on BDI model.

Jason [6] is a Java-based interpreter for an extended version of the AgentSpeak language. An AgentSpeak agent is defined by means of a set of plans that the agent is able to execute in certain situations. An AgentSpeak plan is defined as follows:

$$+triggering - event : context < -body$$

The **Triggering Event** describes the situations in which a plan may be applicable for execution. The **context** can be used to specify the condition to make the plan applicable even if an event has triggered that plan. The **body** can be considered the consequent of the event linked to the context. Within the body commonly are defined the actions that agent must perform to fulfill its own goals.

C. J-Moise+

J-Moise+ [5] is an implementation of the Moise+ organizational model. J-Moise+ is based on Jason and consists of both an OrgBox Api and a special agent called OrgManager. Agents use the OrgBox Api to access to the organizational layer. While the OrgManager stores the current state and maintains the consistency of the Organizational Entity during its life-cycle. J-Moise+ basically offers a set of actions to change the state of the organization and produces some events related to organizational changes to which the agent can react.

III. The Proposed Notation

A detailed description of UML is out the scope of this paper, we here define only the constructs used to model organizations with Moise+. In the following subsections, we describe three kinds of diagram applied to the classical example ("Writing Paper") reported in the Moise+ Tutorial [10].

A. Organizational Diagram

The Moise+ structural specification defines the available roles, groups, and relations between these within the MAS organization. Using the normative specification we can constrain the agents behavior by specifying what missions an agent ought to follow and what missions an agent is allowed to follow when playing certain roles.

We use a UML class diagram (named *Organizational Diagram* OD) for representing the structural and normative specification. The *Organizational Diagram* focuses on Moise+ elements such as Group, Roles, Missions and different kinds of relationships.

The development methodology of an organization with Moise+ is out of the scope of this paper, but in order to understand this diagram we can say that the building of this diagram is subdivided in two phases. During the first phase all elements concerning the Moise+ structural specification are established, while the second phase starts at the end of the definition of the schema structural diagram and it aims to define the norms the agents should obey when they adopt a role. Figure 1 illustrates the graphical representation of organizational diagram elements.

Groups - A group is represented by means of a package with little men icon. It may contain several structural elements (Roles) and other grouping elements (sub-groups). The root group represents the entire organization.

Roles - A role is an UML class depicted as a little man. Its properties are represented in the form of class attributes. Roles can be logically related to one another using associations. An abstract role, instead, is identified using an italic font.

Relationships - Model elements are related each others with dependencies, associations and generalizations. A *dependency* is a generic relationship, indicating that an element depends in some way on another. A *generalization* specifies a relationship between roles in which specialized roles inherit features of the general role. An *association* describes a link between elements of the Moise+ model. We used UML stereotypes in order to attribute a semantic of the association with Moise+ domain-specific concepts.

In particular **Moise+ organizational links**, defining the way in which social exchanges between agent roles occur, are represented by means of associations between roles labeled with an *Authority*, *Acquaintance* or *Communication* stereotype.

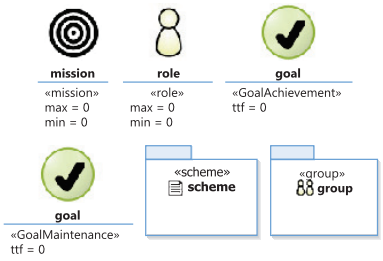

Fig. 1. The Defined Notation Elements

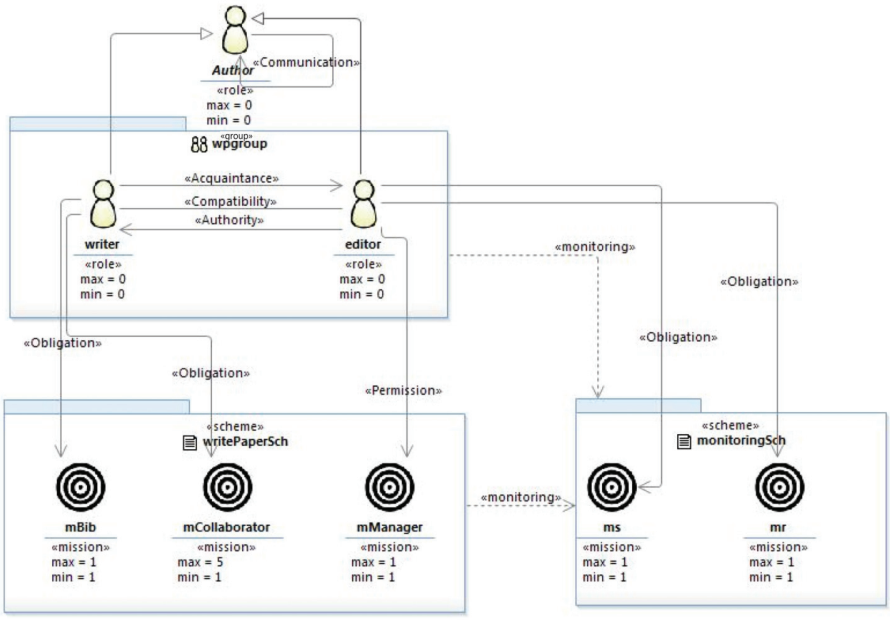

Fig. 2. The Writing Paper Example - OD

Each propriety of a link can be expressed by UML association constraints.

Moise+ compatibility links, instead, are defined by means of associations between roles labeled with the *Compatibility* stereotype. Two roles connected by a bidirectional compatibility link define the possibility of an agent to adopt both roles at the same time. Each propriety of a link can be expressed by UML association constraints.

Figure 2 shows the Organizational Diagram for the Writing Paper organization. In this example, a set of agents aims to write a paper. For this purpose, the Moise+ authors define an organization with one group (*wpgroup*) and two roles (*Writer* and *Editor*). These roles are an extension (represented by means of UML *generalization* in the OD) of the abstract role *Author*. An agent can play several roles only if they are compatible. As exemplified in figure 2, an agent playing the writer role can play the role editor at the same time and vice-versa because they are linked by an *UML compatibility association*.

In the Moise+ model, a role is usually linked by means of norms (*Obligation or Permission*) to one or more missions defined in a particular scheme.

One of the *Writer*'s mission (see Figure 2) is *mbib* (i.e. getting references for the paper). The *norm* linking the mission to the role is an *Obligation*, that is the agent playing the Writer role must commit to this mission. This is shown in the Organizational Diagram through the stereotype *obligation*. The *Editor*, instead, may commit to the mission *mManager* because the link is a *Permission norm*. The association between roles is stereotyped in order to represent organizational links such as *Acquaintance*, *Compatibility*, *Communication* and *Authority*.

B. The Scheme Structural Diagram

The Moise+ Functional Specification deals with the concepts of agents' missions and their global plans. Plan represents the set of goals to be pursued. Plans and missions compose (or are assembled into) the *social scheme*. We use two different views (or models) for representing the elements the functional specification is composed of: the Scheme Structural Diagram (SSD) and the Scheme Functional Diagram (SFD).

The Scheme Structural Diagram allows to model the *social schemes* of the organization through a UML class diagram. The elements of this diagram are:

Goal is represented by a class element reporting the name, the stereotype and the attribute field; each of them corresponds to a specific feature of the Moise+ concept of goal: the class name addresses the goal id. The stereotype represents the two types of goal namely achievement and maintenance. The default type for every goal in Moise+ is achievement but in the SSD the goal type has to be stated in any case. As regard the attribute compartment, it basically contains the *ttf* attribute value prescribing the time requested for fulfilling the goal.

Mission is also represented through a class stereotyped as *mission*. Here the attributes' compartment contains values for the minimum and the maximum commitments to the mission.

The **Social Scheme** is modeled by means of a package where classes (i.e. missions) are grouped in order to represent the social organization of goals and missions. There can be more than one package in a single SSD thus representing the existence of different schemes in the same organization. The package's name corresponds to the social scheme id.

As regard relationships among elements, in this diagram we only use two kinds of relationship: the aggregation and

the dependency; the latter is used for representing how two different schemes depend on each other, the former is used for relating missions and goals. With respect to Moise rationale, goals are aggregated into missions that can be distributed/ committed to agents.

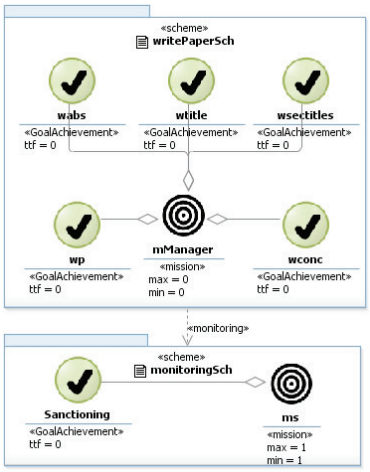

Fig. 3. The Writing Paper Example - SSD

Figure 3 shows a portion of the SSD for the write paper example[1].

In the Scheme Structural Diagram, a Social Scheme is modeled by means of a package containing classes (i.e. missions and goals). Within a package the structural composition of goals and missions is defined. For instance, the SSD for writing paper example is composed by two Social Schemes, *writePaperSch* and *monitoringSch*. The portion of writePaperSch scheme reported in figure 3 shows how the *mManager* mission is a composition of five goals: *wp*, *wtitle*, *concl*, *wabs*, *wsectitles* that respectively aim to write the paper, the title, the conclusion, the abstract and the title of each section. While the illustrated portion of monitoringSch scheme shows *ms* mission formed by only *Sanctioning* goal. In the SSD is also possible to underline the dependences between different social schema. As 3 shown, the social scheme *writePaperSch* is related to the *monitoringSch* scheme through a "monitoring" dependency relationship.

C. The Scheme Functional Diagram

The Scheme Functional Diagram represents the behavioural view of the Moise+ functional specification, it is realized by means of an UML activity diagram and it aims at representing, through a set of associated activities, how a goal can be decomposed in sub-goals. Each activity represents the work done by agent to fulfill the goal. The elements of this diagrams are:

[1]Because of space concerns only portions of diagrams are reported. Complete diagrams can be found in http://www.pa.icar.cnr.it/cossentino/moisenotation/

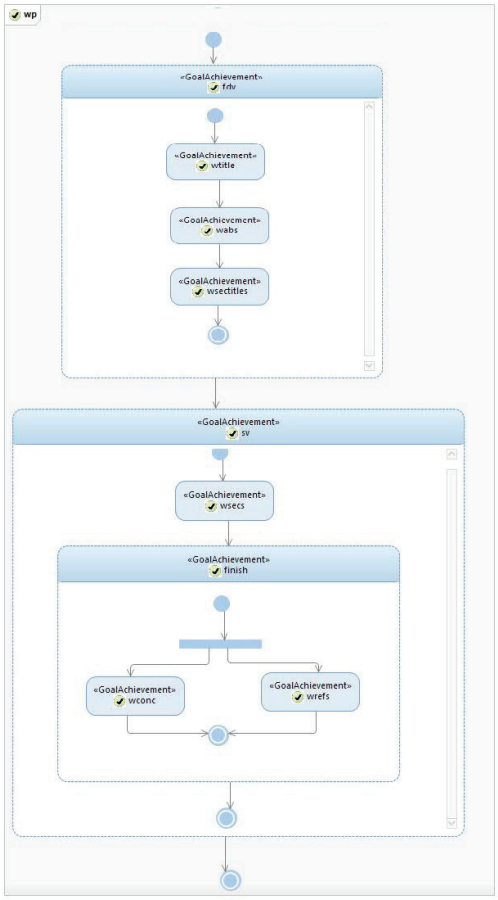

Fig. 4. The Writing Paper Example - writePaperSch SFD

The **Goal** is then represented by an activity where the name is the goal's id and the stereotype represents the type of the goal (achievement or maintenance - see the previous subsection) The Moise+ model allows to decompose goal in sub-goals by means of a plan operator. There are three different kinds of plan operator: *sequence*, *parallelism* and *choice*, the first means that a goal g_i (having two sub-goals $g_{i,i}$ and $g_{i,i+j}$) can be achieved only if the close sequence of $g_{i,i}$ and $g_{i,i+j}$). All of them can be easily represented by means of the UML activity diagram syntax, for instance the parallelism is represented through the fork and the choice through the decision diamond. Sequence is represented by a straight arrow line.

As said before, in this paper our concern is about the notation/models to be used for representing MAS organizations. If we would use them during a design process phase we should consider that we can draw more than one SFD, one for each package (i.e. social scheme) of the SSD.

Figure 4 and figure 5 show the Scheme Functional Diagrams (SFD) built for the *wp* and *monitoring* goals of the Writing Paper organization which are the root goals of *writePaperSch*

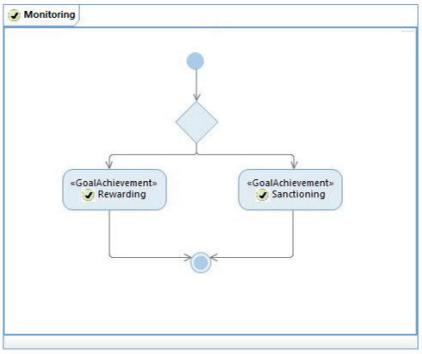

Fig. 5. The Writing Paper Example - MonitorinSch SFD

and *monitoringSch* (defined in the previous section) correspondingly. The SFD of the *writePaperSch* (see figure 4) explains how to achieve the root goal of the scheme. In detail, the fulfillment of the *wp* goal (i.e. write a paper) depends on the achievement of the *fdv* (first draft version) and *sv* (submit version) goal. The *sv* goal is reachable only after that the *fdv* is satisfied. In turn, *fdv* is achieved executing the atomic goals *wtitle*, *wabs* and *wsectitles* sequentially.

It is important to highlight there are three different types of goal execution: sequential, parallel and choice. If two goals are related with a sequential relationship then the goal target can be reached only after that the source goal is reached. If two goals are related with a parallel relationship then both goals can be reached simultaneously. Finally, a choice relationship indicates that it is possible to choose the goal to achieve.

Besides it is important to note that (see 4 and figure 5) the root goal is represented with a box with the goal icon at the top left corner instead of with an activity, this is due to the features of the tool we use for drawing activity diagrams. The concept of goal does not depend on the graphical box they are represented but are related to the specific icon.

IV. Related Works

A proposal for the introduction of groups in MAS modeling has been presented by J. Odell et al. in [14]. The proposed metamodel is based on three main concepts: Agent, Agent Role (Classifier, Assignment) and Group. The peculiarity of this approach is in the presence of the agentified group in opposition to the Non-agentified one. This does not represent an explicit attention for the presence of non-agent-oriented entities in the application. Conversely, non-agentified groups are composed of agents just like the others, but they are not addressable as an agent entity (i.e. the group does not exhibit the usual properties of an agent). Within a group, agents interact according to the roles they play.

As regards the comparison of this work with the notation we presented in this paper, the authors of [14] present very limited examples of notation. Mainly, a group is represented in a form that resembles the UML class without the operation and attribute compartments. Purposefully, the author avoid to propose notations for agents, roles and the other elements of their proposed metamodel.

Remaining in the context of AUML-related researches, it is worth to remind the long work done by the FIPA Modeling Technical Committee and its members at the beginning of years 2000. In this context several proposals arose. Among the others, Parunak and Odell presented in [15] some ideas for the representation of social structures and relationships.

They introduced swimlanes in class diagrams in order to partition the diagram in zones representing groups. Within each organization the diagram may depict roles and the agents playing them. Another proposed diagram was concerned with the description of the dynamic behavior of agents/roles in terms of their interactions. Essentially it is an extension of the UML 1.0 sequence diagram containing some notation elements that have been introduced in following versions of UML.

A more extended notation has been proposed by L. Padgham et al in [16]. This notation has been conceived with the aim of supporting most of the existing AOSE methodologies.

An interesting point of this work is that the authors defined a notation leaving a large margin for the definition of the semantics that is behind that. In this way, the notation may be easily ported to support different approaches. More in details, it has been applied to O-MASE, PASSI, Prometheus and Tropos (partially). The notation includes graphical icons for representing almost all the elements of an agent-oriented design, organizations included (agent, role, position, goal, ...).

From this point of view, the notation presented in [16] is more complete than the notation we propose in this paper.

However, the authors in [16] present several diagrams, none of them behavioral. This is a relevant difference with the work presented in this paper. In fact, our notation also includes the Scheme Functional diagram that is a behavioral representation of the system.

INGENIAS [17][18] is a framework for developing MASs offering to designers the possibility of following the workflow of the methodology also with the aid of the tool (INGENIAS Development Kit IDK). The tool supports a specific notation for representing the abstractions on which INGENIAS allows to develop MASs.

INGENIAS is suitable for modelling and developing MASs with the following main abstractions: agent, task, role, organization, and goal. Modelling with INGENIAS implies basically using the Unified Software Development Process (USPD) [19]; each phase/iteration aims at developing different models or viewpoints on specific aspects of the MAS under development. For the sake of the work proposed here we are interested in considering the Organizational Viewpoint.

The Organizational viewpoint describes the environment where agents live and interact each other by means of resources and tasks in order to pursue goals. Modelling the Organization is done by dividing the MAS into *groups* and *workflows* where all the involved entities are related by aggregation and inheritance relationships; roughly speaking, in INGENIAS, groups give the mean to identify subsystems interacting through workflows.

Each element of the Organization viewpoint has a precise notational counterpart, for instance the goal is represented by a circle and the group in a box with two kinds of head. This allows to model how the organization is divided in groups (each group again can be decomposed in groups) made by agents that play roles. From a modelling point of view it is important to note that the goal is associated to the organization and that in INGENIAS the concept of agent is central and is related to the concept of role whereas in MOISE+ the central element group and the role. This logical difference can be found in the two notations and in the related diagrams we can draw; anyway both of them allow representing the whole portion of metamodel including organizational concept but INGENIAS does not provides means for representing norms.

Tropos [20] is an agent-oriented software engineering methodology mainly based on the notion of goal.

The Tropos methodology is articulated in four different phases from the requirements analysis to the agent system implementation. The requirements analysis covers two phases: the *Early Requirements Analysis* phase, concerning with the studying of the problem, produces an organizational model and the *Late Requirements Analysis* phase where the system-to-be is described. The agent system implementation is performed through the *Architectural Design* phase, where the system architecture is defined, and the *Detailed Design* phase where all system components are specified. During the first two phases an actor and a goal diagram are produced.

An *Actor Diagram* is a graphical representation of the application domain stakeholders, their objectives and dependences. A *Goal Diagram* is a refinement of the previous diagram underlining the goals of a single actor. These two diagrams show essentially five concepts: Actors, Goals, Resources, Tasks and Dependences. The *Actors* are the intentional entities such as agents (software or human), roles (abstract representation of behaviors within some specialized domain) or position (set of roles typically played by an agent). *Goals* are the objectives of an actor, divided into hard goals and soft goals. *Tasks* are the way to achieve a goal. *Resources* are means used by agent in order to reach their goals. Finally, *Dependences* are relationships between actors.

The notation used in the Tropos diagrams in order to represent the above elements is very simple. An actor is depicted by means of a circle, its goals are ovals and its softgoals are clouds shape. The used resources and tasks are represented as rectangles and hexagons respectively. The dependences between actors are arrows with a specific content. This content represents the *dependum* (i.e. goal, task or resource) that is the element through which two actors depend each other.

Tropos does not support natively the concept of organization. Its authors have proposed organizational patterns [21] in order to facilitate the construction of organizational models. These patterns are defined from real world organizational settings, such as Joint Venture, Pyramid, Flat Structure and many others [22], and formalized using the Tropos notation.

V. Discussions and Conclusions

In the field of agency, the complexity of current systems and applications led to an increasing number of agents employed in the multi agent system that must expose autonomous and organizing capabilities also for substituting and making decisions on the behalf of user.

One important topic in these kind of systems is how to manage the agents by creating organizations in the same way the human, and more generally, the biological systems do.

The design and the implementation of organization in MASs is related to this topic. Our work concerns the creation of a design process for developing MASs organized in hierarchical structures, such as holons, that can be implemented with J-Moise+. In this paper we present the first step of this ongoing work: the UML-based notation to be used for representing organizational elements in the design process work products.

We created a notation enabling us to model organization through three different artefacts where all the elements of Moise+ organizational model are represented. One of the most important results is that we eliminated the difficulty related to the use of the predicative form for representing goal, norms, etc. Moreover we obtained the capability of converting the work product, the diagram, in a *xmi* file and then through an easy transformation in a *xml* file thus directly obtaining Moise+ code or better, if necessary, any other kind of code.

Acknowledgment

This work has been partially supported by the EU project FP7-Humanobs and the IMPULSO project funded by the Italian Ministry for Economic Development.

Authors would like to thank Paolo Giorgini for his useful comments and suggestions.

References

[1] M. Wooldridge and N. Jennings, "Intelligent Agents: Theory and Practice," *The Knowledge Engineering Review*, vol. 10, no. 2, pp. 115–152, 1995.

[2] M. J. Wooldridge, *Introduction to Multiagent Systems*. John Wiley & Sons, Inc. New York, NY, USA, 2001.

[3] V. Dignum and F. Dignum, "Modelling agent societies: co-ordination frameworks and institutions," *Progress in Artificial Intelligence*, pp. 7–21, 2001.

[4] J. F. Hübner, J. S. Sichman, and O. Boissier, "Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels," *International Journal of Agent-Oriented Software Engineering*, vol. 1, no. 3, pp. 370–395, 2007.

[5] J. F. Hübner, "J-moise+ programming organizational agents with moise+ and jason (2007)."

[6] R. H. Bordini, J. F. Hübner, and M. J. Wooldridge, *Programming multi-agent systems in AgentSpeak using Jason*. Wiley-Interscience, 2007.

[7] F. Zambonelli, N. R. Jennings, and M. Wooldridge, "Developing multi-agent systems: The Gaia methodology," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 12, no. 3, pp. 317–370, Jul. 2003.

[8] M. Cossentino, N. Gaud, V. Hilaire, S. Galland, and A. Koukam, "ASPECS: an agent-oriented software process for engineering complex systems," *Autonomous Agents and Multi-Agent Systems*, vol. 20, no. 2, pp. 260–304, 2010.

[9] K. Fischer, M. Schillo, and J. Siekmann, "Holonic multiagent systems: A foundation for the organisation of multiagent systems," *Holonic and Multi-Agent Systems for Manufacturing*, pp. 1083–1084, 2004.

[10] J. F. Hübner, J. S. Sichman, and O. Boissier, "Moise tutorial. (for moise 0.7)." [Online]. Available: moise.sourceforge.net/doc/tutorial.pdf

[11] ——, "Moise+: towards a structural, functional, and deontic model for mas organization," in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*. ACM, 2002, p. 502.
[12] F. Bellifemine, G. Caire, and D. Greenwood, *Developing multi-agent systems with JADE*. Wiley, 2007.
[13] A. Rao, "AgentSpeak (L): BDI agents speak out in a logical computable language," *Agents Breaking Away*, pp. 42–55, 1996.
[14] J. Odell, M. Nodine, and R. Levy, "A metamodel for agents, roles, and groups," *Agent-Oriented Software Engineering V*, pp. 78–92, 2005.
[15] H. Van Dyke Parunak and J. Odell, "Representing social structures in uml," *Agent-Oriented Software Engineering II*, pp. 1–16, 2002.
[16] L. Padgham, M. Winikoff, S. DeLoach, and M. Cossentino, "A unified graphical notation for aose," *Agent-Oriented Software Engineering IX*, pp. 116–130, 2009.
[17] J. Pavòn, J. J. Gòmez-Sanz, and R. Fuentes, "The INGENIAS methodology and tools," in *Agent Oriented Methodologies*. Idea Group Publishing, 2005, ch. IX, pp. 236–276.
[18] INGENIAS, "Home page," `http://grasia.fdi.ucm.es/ingenias/metamodel/`.
[19] I. Jacobson, G. Booch, and J. Rumbaugh, *The unified software development process*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1999.
[20] P. Giorgini, M. Kolp, J. Mylopoulos, and J. Castro, "Tropos: A requirements-driven methodology for agent-oriented software," in *Agent Oriented Methodologies*, ch. II, pp. 20–45.
[21] M. Kolp, P. Giorgini, and J. Mylopoulos, "Organizational patterns for early requirements analysis," in *Advanced Information Systems Engineering*. Springer, 2010, pp. 1030–1030.
[22] A. Fuxman, P. Giorgini, M. Kolp, and J. Mylopoulos, "Information systems as social structures," in *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*. ACM, 2001, pp. 10–21.