

Issues for Organizational Multiagent Systems Development

Emilia Garcia

Department of Information
Systems and Computation
Technical University of
Valencia
Valencia, Spain
mgarcia@dsic.upv.es

Adriana Giret

Department of Information
Systems and Computation
Technical University of
Valencia
Valencia, Spain
agiret@dsic.upv.es

Estefania Argente

Department of Information
Systems and Computation
Technical University of
Valencia
Valencia, Spain
eargente@dsic.upv.es

Vicente Botti

Department of Information
Systems and Computation
Technical University of
Valencia
Valencia, Spain
vbotti@dsic.upv.es

ABSTRACT

Organizational multiagent system (OMAS) are rapidly emerging as a powerful paradigm for developing complex systems. Their development process implies specific requirements and software engineering tools. Recently a great number of methods and frameworks to develop OMAS have appeared. Each of them offers different functionality and they have distinct characteristics and perspectives. The main contribution of this paper is a comprehensive list of fundamental OMAS development issues. These issues will be a starting point to define a complete list of OMAS development requirements. From these requirements it could be possible to define an evaluation framework for OMAS development tools in order to help the developers in evaluating OMAS tools and applications.

Keywords

Multi-agent systems, MAS organizations, software engineering

1. INTRODUCTION

Organization multiagent systems (OMAS) have been successfully employed as a paradigm for developing agent systems [7, 17]. One of the advantages of organization development is that systems are modeled with a high level of abstraction, so the conceptual gap between real world and models is reduced. Also this kind of systems offers facilities to implement open systems and heterogeneous member participation [25].

OMAS development implies new requirements on traditional MAS models and technology, including the integra-

tion of organizational and individual perspectives and the dynamic adaptation to environmental changes [10]. Therefore, traditional MAS software engineering is not enough to develop OMAS. They need software engineering methods and frameworks that cover organizational concepts and offer the necessary technology. In the last years several OMAS methodologies [4], modeling techniques [22] and platforms [5, 15] have been developed. However, each approach uses its own terminology and offers different functionality, so it is very difficult for developers to choose between one or another. Many designers have doubts about how to translate organizational concepts into final execution entities for an application [6] and what platform and implementation requirements are needed. There is no common agreement on what should be the complete list of these "new" requirements for organization-oriented MAS systems. Moreover, there is no evaluation framework that helps in determining the correctness or completeness of a given OMAS method, tool or execution platform for this kind of system.

Some works like [13, 28] analyze and compare different agent-oriented frameworks to develop MAS. They show a list of the most important features that are needed to develop MAS and propose them to evaluate current methods and MAS development kits. However, this list of features is not complete enough for evaluating OMAS, because it does not deal with organizational concepts and related technology required for developing OMAS. The main goal of this paper is to contribute with a comprehensive list of fundamental OMAS development issues. These issues will be a starting point to define a complete list of OMAS development requirements. From these requirements it could be possible to define an evaluation framework for OMAS development tools in order to help developers in evaluating OMAS tools and applications.

In order to define the fundamental OMAS development issues we have made a detailed study of the state of the art methods and tools for OMAS. Section 2 overviews a set of the most representative ones. In this study we have analyzed their organizational concepts and technology associ-

Jung, Michel, Ricci & Petta (eds.): *AT2AI-6 Working Notes, From Agent Theory to Agent Implementation, 6th Int. Workshop*, May 13, 2008, AAMAS 2008, Estoril, Portugal, EU.

ated. From this study we propose the list of fundamental OMAS development issues detailed in Section 3, we also present a brief discussion on the importance of these issues in OMAS development. Finally, Section 4 presents some conclusions and future work.

2. STATE OF THE ART OMAS DEVELOPMENT TOOLS

In this section we briefly present the most representative set of state of the art development tools for OMAS. We have studied these tools in order to figure out the OMAS fundamental issues required and already provided by them. The goal of this section is just to present an introduction of them. The detailed information on each tool can be found in the referenced literature. In the next section we present the insights of the OMAS development issues related with these tools.

AGR (Agent, Group, Role) [17] is an organizational methodology that is the evolution of the Aalaadin model [16]. In AGR agent, group and role are the primitive concepts. An agent is an active, communicating entity playing (multiple) roles within (several) groups.

The Tropos methodology, in the latest version [20], adopts an organizational viewpoint and explicitly studies the identification of the organizational structure. It proposes using generic multiagent structures that are based on human organizations.

GaiaEXOA (Gaia Extended with Organizational Abstractions) [31] is an evolving extension of the Gaia methodology for designing open MAS.

Ingenias [26] is a methodology for the development of OMAS that is supported by an integrated set of tools, the INGENIAS Development Kit (IDK). It integrates results from research in the area of agent technology with a well-established software development process, which in this case is the Rational Unified Process (RUP).

Electronic Institutions (EI) [1] is a framework that is focused on the design and implementation of an open MAS. It allows the definition of social norms and agent behavioral control. This framework provides several tools to define and implement electronic agent institutions.

Organizational Model for Normative Institutions (Omni) [11] is an integrated framework for norms, structure, interaction and ontologies for modeling organizations in MAS. It is a unification of two other models: the OperA and the HarmonIA framework.

Moise+ is an organizational model for Multi-Agent Systems based on notions like roles, groups, and missions. Moise+ models may be implemented in any platform. Authors present some software to work with Jason, Saci and S-Moise+ [23] which is developed by Moise+ authors.

Jack Teams [27] is an extension to JACK Intelligent Agents platform that provides a team-oriented modelling framework. As Jack, it is based on the BDI agent model and it is implemented in Java.

3. OMAS DEVELOPMENT ISSUES

OMAS software engineering is based on agent-oriented software engineering (AOSE). However, AOSE is not enough for developing this kind of systems as it does not consider the own characteristics of organization concepts and techniques [4].

The aim of this section is to show the most important software engineering issues for OMAS development. Because of the different perspectives and the terminological and conceptual confusion in OMAS [8], the identification of a set of independent, orthogonal features which completely characterize the OMAS development process seems infeasible. The purpose of this section is to make an overall analysis of the needs that arise when developing OMAS, relying more on the concepts than in a specific terminology. These issues are classified into three categories: (i) methodology and modeling language; (ii) development tool; and (iii) execution platform. Following, those categories are deeply discussed.

3.1 Methodology and modeling language

A well-defined methodology greatly helps developers to move from the requirements of the application to the final built system. A complete analysis of the most important features in agent methodologies can be found in [13]. This section is focused on the issues that must be considered when MAS organizational concepts are taken into account. Specifically, methodological features that need to be analyzed are:

- **Development process:** the methodology and the modeling language should be able to extract the organizational requirements; and model the organizational concepts and structure, and also the individual behaviours and objectives of agents. An analysis whether they cover the whole development process is needed. Most approaches, such as AGR, Roadmap and Tropos, only cover the analysis and the design stages. Despite this, approaches, such as Electronic Institutions, Ingenias and Omni, also cover the implementation stage.
- **Methodology concepts and modeling language relationship:** The gap between the concepts of the methodology and the modeling language should be as little as possible, so a complete organizational modeling language is needed [6]. This modeling language can be formal or informal. Most part of current OMAS methodologies offer an informal modeling language that cover completely the concepts and the relationships studied in the methodology. Despite this, there are approaches that introduce formal modeling. For example, all dimensions of Omni have a formal logical semantics, which ensures consistency and possibility of verification of the different aspects of the system.
- **Social patterns:** Patterns describe a problem commonly found in software design configurations and prescribe a flexible solution for the problem, so as to ease the reuse of that solution. This solution is repeatedly applied from one design to another, producing design structures that look quite similar across different applications [12]. They are very useful and reduce the developing time, but only few approaches integrate them. One of them is Tropos that integrates few social patterns in the organizational topologies. In [12] some of these patterns are explained (broker, matchmaker, mediator, monitor, embassy, wrapper, contract-net).
- **Domain dependence:** Methods should offer domain-independent organizational representations and domain-specific frameworks to capture particular relevant organizational characteristics [21].

On the other hand, there are specific organizational concepts that must also be modelled by an OMAS methodology, based on five dimensions [8, 4]: structural, dynamic, functional, normative and environment.

3.1.1 Structural Dimension.

It represents the structure of the organization and all the elements that persist in the organization independently of their members. The structure is defined by its roles, groups, and their dependencies and links. The issues related with the structural dimension are:

- *Topology Selection:* There are many organizational topologies (for example Tropos propose Flat-Structure, Pyramid Style, Chain of Values, Matrix, Structure-in-Five, Co-optation, Joint Venture, Bidding, Arm's Length y Hierarchical Contracting). Which topologies are supported by the methodology and by the modeling language is an important development issue. Most complete methodologies should provide a **guideline** to help developers in the decision of which structure is more appropriated. For example, Tropos offers some guidelines to determinate which is the most appropriate topology for each application. These guidelines are based on general aspects of the performance (fault tolerance, adaptability, coordinability) and they do not take into account organizational components like departments.
- *Composed organization:* This concept considers whether the approach allows the creation of an organization inside another organization. The most part of the methodologies allow it (AGR, GaiaEXOA, Moise+, etc.). Also whether an organization can be created from other pre-existing organizations.
- *Social relationships:* Methodologies might take into account and enable modeling role dependencies and the type of social relationship between the agents and with organizations [10]. Some role dependencies are: heritage, communication, compatibility, coordination, authority or power control. Some of relationship types between agents and organizations are: knowledge links (who can get information about other agents); communication links (who can interact with); authority links (who has control over others); etc.

3.1.2 Dynamic Dimension.

It represents the evolution of the organization, i. e., how agents go inside and outside of the organization and how they change their roles depending on their capabilities and objectives in each moment. The following concepts are needed:

- *Dynamical models:* The dinamicity of the system should be modeled, i. e., the modeling language should be able to represent how agents go in and out of the organization, how they change their roles and how the organizations are created and/or destroyed. Each approach represents the dinamicity of the system in a different way and given different functionality. For example: AGR models how agents change from one group to another. Omni uses contracts to specify when an agent can have a given role. Furthermore, approaches like EI or Moise include an entity which checks that the actions are valid before executing them.

- *Interactions:* The possibility to have open and dynamic systems that allow the interaction of heterogeneous agents is a desirable feature. To get this the definition of a *common ontology* is necessary. Also communicative acts should follow **interaction patterns** or **protocols**. It is necessary to define a set of the **valid illocutions** that agents can exchange and that satisfies a common ontology, a common communication language and knowledge representation language [2]. The most part of the approaches define interaction protocols, but only few of them, such as EI and Ingenias, are able to model a common ontology that completely specify the valid illocutions.
- *Context:* Some approaches specify the situations or context in which the organizations can be during their execution, and how an organization changes between one situation to another. The context specifies the state of the agents, which roles they play and the established norms [1]. Despite this is a very interesting feature, only few approaches like EI and Moise-Inst [18] (an extension of Moise) define contexts.

3.1.3 Functional Dimension.

It represents the organization goals and each component goals. Also it indicates how these goals are achieved, i. e., their decomposition in tasks and plans. Finally, it defines which functionality is offered by the organization and agents. In this case, the most relevant concepts to be taken into account are:

- *Goals:* It is important to not forget that agents which form an organization are autonomous and they have their own goals, behaviours and beliefs in addition to those from the organization. For that reason the approach should be able to model **individual goals**, global or **organizational goals** and how these global objectives are **decomposed** into individual goals in order to be achieved. The most part of the approaches support this feature.
- *Goal decomposition:* The decomposition of goals into tasks and plans should be modeled. Also the most part of the approaches support this feature.
- *Functionality:* The offered behaviour of any OMAS entity should be specified. Currently, there is no approach that specifies which functionality is offered by an organization.

3.1.4 Normative Dimension.

It represents the set of norms that control the organization. Norms facilitate the mechanisms to drive the behaviour of agents, specially in those cases when their behaviour affects other agents [24].

In organization-oriented methodologies two different trends can be observed when comparing several approaches [4]. On the one hand, methods such as Ingenias [26]. Agent-Group-Role [17] detail system roles, groups and relationships but they do not explicitly consider social norms. On the other hand, other methods and frameworks, such as Electronic Institutions [14], are focused on the social norms and explicitly define control policies to establish and reinforce them. Moreover, an extension of AGR to support norms is presented in [19].

Many kinds of norms can be distinguished, but not all methodologies and modeling languages allow the specification of all of them. Some of these types of norms are [24]: (1) Deontic (Obligations, Permissions and Prohibitions); (2) Legislatives, for creating, modifying or revoking norms; (3) Reinforcement, for controlling and penalizing; (4) Rewards.

3.1.5 Environment Dimension.

It represents all the elements of the environment that interact with the organization. The following concepts are needed:

- *Resources*: The mechanism to access resources (read, interact, modify, etc.) should be modeled [31, 26]. Some methodologies like Roadmap, GaiaEXOA and Omni, model the resources, their contextual relationships and how these resources are accessed.
- *Perceptors and Effectors*: Some approaches like [29] model observations as the ability of an entity to perceive the state of (or to receive a signal from) an observed entity by means of perceptors. Perceptor types are used to specify (by means of perceiving acts) the observations that agents can make. The specification of which entities can observe others is modeled with a **perceives dependency**. Different aspects of effecting interactions are modeled analogously, by means of **effectors**, effector types, effecting acts, and effect dependencies.
- *Stakeholders*: They represent the interaction links of the organization with its environment, detailing who takes benefit of the organizational results or who does the organization depend on [30]. Only few approaches model stakeholders, one example is Omni that model the entities that take benefit of the organization or that need it. Also identify which are their objectives and their dependences on the organization.

3.2 Development tool

Development tools are usually divided in two different kind: the specification tool that allows modeling the system, and the implementation tool that allows implementing the final code of the application.

Some tools try to integrate or connect both parts [1]. They add automatic code generation techniques that reduce significantly the implementation time and errors. Despite this, nowadays the gap between the model and the implementation is very high [28] and the most common situation is that a big part of the concepts defined in the models cannot be directly translated to the final implementation. Currently there is no tool that completely integrates the model and the final code.

Only the requirements that appear when the organizational concepts are being added to a MAS are taking into account in this paper. A complete analysis of the most important features of traditional MAS development kits can be found in [13].

3.2.1 Modeling tool.

It should offer modeling facilities by using the selected organizational modeling language. It should cover completely the modeling language and the methodology. It is convenient that some or all the guidelines offered by the method-

ology are integrated with the modeling tool, but current systems do not offer this facility.

3.2.2 Implementation tool.

The implementation tool should integrate the entire range of modeling features and should ease the translation from these modeling features to the corresponding execution elements of a given agent execution platform.

Lots of methodologies and software engineering works only offer a theoretic analysis and lots of them do not provide any development tool. Two of the most complete development tools are:

EI offers : Islander (a graphical tool that supports the specification and verification of the institutional rules); Simdei (a simulation tool to animate and analyze Islander specifications prior to the deployment stage); aBuilder (an agent development tool which given an Islander specification supports the generation of agent skeletons for that institution); Ameli (a software platform to run institutions); Monitoring tool (a tool which permits the monitoring of EI executions run by Ameli). Nevertheless, there are important lacks in EI tools: the model tool only takes into account organizational concepts; the development tool (aBuilder) only automatically generate agent skeletons and the code agent should be manually completed.

On the other hand, Ingenias is supported by an integrated set of tools, the Ingenias Development Kit (IDK). These tools include an editor to create and modify MAS models, and a set of modules for code generation and verification of properties from these models. This approach covers the entire development process in a basic way, but, it has important lacks in the transformation from models to the final implementation. It only offers a basic generation of code skeletons and does not provide an implementation environment.

3.3 Execution platform

Regarding agent platforms, the most well-known agent platforms (like Jade) offer basic functionalities to agents, such as AMS (Agent Management System) and DF (Directory. Facilitator) services; but designers must implement nearly all organizational features by themselves, like communication constraints imposed by the organization topology [3].

- **Organization representation**: Organizations can be materialized in the following ways [6]: (1) developers does not define the organization structure, although the observer can see an emergent organization; (2) the organization exists as a specified and formalized schema, made by a designer but agents do not know anything about it and do not reason about it; (3) each agent has an internal and local representation of cooperation patterns which it follows when deciding what to do; (4) agents have an explicit representation of the organization which has been defined. Thus an agent is able to reason about it and uses it in order to initiate cooperation with other agents . A good example of the 4 classification may be S-Moise+ or Ameli (EI platform). They have an explicit representation of the organization and both have similar architectures. They follow a three-layer architecture: the application layer that is formed by the autonomous agents of the application; the social layer that ensures that the interac-

tion follow the established norms; the communication layer that allow the communication between agents. Application agents are responsible for achieving organizational goals and using the agent proxy offered by the organization to interact with it.

The implementation tool should integrate the entire range of modeling features and should ease the translation from these modeling features to the corresponding execution elements of a given agent execution platform.

- **Control mechanisms:** The platform should have control mechanisms that ensure the satisfaction of the organizational constraints. The most common architectures use **middlewares** between agents or between agents and the organization [15]. This middleware forces agents to respect the constraints of the organization. Also, this middleware allows that heterogeneous agents interact and that the organization changes dynamically. This feature is well supported by Ameli and S-Moise+. They act as a middleware between agents and the communication layer. Each agent has associated a proxy agent offered by the organization which control that all the norms and constraints are validated before the interaction.
- **Description of the organizations:** The organization should have an available description in a standard language. It allows external and internal agents to get some information about the organization at run-time. This last feature is not only useful in open systems, but also when considering a reorganization process. A good example of specification of the organization and its benefits can be found in the S-Moise+ platform.
- **AMS and DF extension:** The AMS and the DF offered by traditional MAS platforms should be improved. The AMS should have the information of the existing organizations and their members. The DF should publish the services offered by the agents individually and the services offered by an organization. It should have not only the name of the service offered, but also a description of it to allow open systems.
- **Communication layer:** The kind of communication layer used in the communicative acts is a very important feature. Some of them, such as FIPA-ACL (used by Ameli) and KQML (used by S-Moise+), are more suitable for open systems than TCP/IP, CORBA and RMI [6].
- **Monitorization:** The platform should offer a mechanism to monitorize the state of the agent and of the organizations.
- **Modeling concepts support:** The platform and the programming language should cover all these concepts (explained in Section 3.1). For example, which types of topologies support the platform, which kind of norms, etc. are very interesting features to analyze. No all the platform has a complete modeling concepts support, for example Ameli is focused on the management of rules and norms but do not support the definition of complex topologies. Jack Teams allows the creation of composed "Teams" but it do not offer support for other topologies.

- **API:** The platform should offer an API that allows [9, 5] to create, destroy and modify organizations; consult and modify the organization description, add, query and delete the agents of an organization; send messages to a whole organization, etc.

3.4 Discussion

OMAS development is a very complex task that implies new requirements on all the stages of MAS development process. As is shown, new methodologies, modeling languages, developing environments and platforms are needed. All these development tools requires specific issues to cover organizational characteristics.

The entire development process should be supported by OMAS software engineering tools. Generally, there are big gaps in the development process. The methodology defines concepts that are not supported by the modeling tool. Also, the modeling tool specifies some entities that do not have a corresponding implementation artifact. Moreover, there are few development tools that automatically generate implementation code for a given execution platform.

The study that we present in this paper, shows that there is no development tool that completely cover all the fundamental OMAS development issues. Furthermore, developers do not have any help to choose between one or another development tool. For this reason, we are convinced that there is a fundamental need of a complete evaluation framework in order to get a qualitative and quantitative measurement of the completeness or correctness of a given OMAS development tool.

4. CONCLUSIONS AND FUTURE WORKS

In this work we have presented a comprehensive list of OMAS development issues. These issues were defined from a detailed study of the state of the art development methods and taking into account the new characteristics of OMAS related to traditional MAS. It is well known that OMAS are specially suited for open and large systems in which organizational structures are required in order to manage the system complexity. These facts makes compulsory to use Software Engineering principles, methods and techniques in the entire development cycle of this kind of systems. Many research efforts have been developed in this field. In this work we have shown that many of the fundamental OMAS issues are not completely covered by these works. Moreover, there is a fundamental need to have evaluation frameworks in order to get a qualitative and quantitative measurement of the completeness or correctness of a given OMAS development tool. With such a framework a developer could select the more appropriate tool for the particular system to develop. The fundamental OMAS development issues presented in this work are a starting point to develop a complete requirement list for OMAS development. From this requirement list it should be possible to define a complete evaluation framework for OMAS development tools.

We are working on the definition of the OMAS development requirement list integrated with traditional MAS requirement, and issues from Service-Oriented MAS. The final goal of our research is the definition of a general qualitative and quantitative evaluation framework for MAS.

5. ACKNOWLEDGEMENTS

This work is partially supported by the PAID-06-07/3191, TIN2006-14630-C03-01 projects and CONSOLIDER-INGENIO 2010 under grant CSD2007-00022.

6. ADDITIONAL AUTHORS

7. REFERENCES

- [1] J. Arcos, M. Esteva, P. Noriega, J. A. Rodríguez, and C. Sierra. Environment Engineering for Multi Agent Systems. *Journal on Engineering Applications of Artificial Intelligence*, 18:191–204, 2005.
- [2] J. L. Arcos, P. Noriega, J. A. Rodríguez-Aguilar, and C. Sierra. E4mas through electronic institutions. In D. Weyns, H. Parunak, and F. Michel, editors, *Environments for Multiagent Systems III*, volume 4389 of *Lecture Notes in Artificial Intelligence*, pages 184–202. Springer-Verlag, 2007.
- [3] E. Argente, A. Giret, S. Valero, V. Julian, and V. Botti. Survey of MAS Methods and Platforms focusing on organizational concepts. In Vitria, J., Radeva, p. and Aguilo, I, editor, *Recent Advances in Artificial Intelligence Research and Development*, Frontiers in Artificial Intelligence and Applications, pages 309–316, 2004.
- [4] E. Argente, V. Julian, and V. Botti. Multi-agent system development based on organizations. *Electronic Notes in Theoretical Computer Science*, 150:55–71, 2006.
- [5] E. Argente, J. Palanca, G. Aranda, V. Julian, V. Botti, A. García-Fornes, and A. Espinosa. *Supporting Agent Organizations*, pages 236–245. 2007.
- [6] O. Boissier, J. F. Hübner, and J. S. Sichman. Organization oriented programming: From closed to open organizations. *Engineering Societies in the Agents World VII*, 4457/2007:86–105, 2007.
- [7] O. Boissier, J. Padget, V. Dignum, G. Lindemann, E. Matson, S. Ossowski, J. Sichman, and J. Vazquez-Salceda. *Coordination, Organizations, Institutions and Norms in Multi-Agent Systems*, volume 3913 of *LNCS (LNAI)*. 2006.
- [8] L. Coutinho, J. Sichman, and O. Boissier. Modeling Organization in MAS: A Comparison of Models. In *First Workshop on Software Engineering for Agent-oriented Systems*, pages 1–10, 2005.
- [9] N. Criado, E. Argente, V. Julian, and V. Botti. Organizational services for spade agent platform. In *IWPAAMS07*, 2007.
- [10] V. Dignum and F. Dignum. A landscape of agent systems for the real world. Technical report 44-cs-2006-061, Institute of Information and Computing Sciences, Utrecht University, 2006.
- [11] V. Dignum, J. Vazquez-Salceda, and F. Dignum. Omni: Introducing social structure, norms and ontologies into agent organizations. *LNAI 3346*, 2005.
- [12] P. A. Do TT, Kolp M. Social patterns for designing multi-agent systems. In *Proceedings of SEKE-2003*, 2003.
- [13] T. Eiter and V. Mascardi. Comparing environments for developing software agents. *AI Commun.*, 15(4):169–197, 2002.
- [14] M. Esteva, J. Rodríguez-Aguilar, C. Sierra, J. Arcos, and P. Garcia. *On the Formal Specification of Electronic Institutions*, pages 126–147. Lecture Notes in Artificial Intelligence 1991. Springer-Verlag, 2001.
- [15] M. Esteva, B. Rosell, J. A. Rodríguez, and J. L. Arcos. AMELI: An agent-based middleware for electronic institutions. In *In Proc. of AAMAS04*, pages 236–243, 2004.
- [16] J. Ferber and O. Gutknecht. A meta-model for the analysis and design of organizations in multi-agent systems. In *Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS'98)*, pages 128–135. IEEE Computer Society, 1998.
- [17] J. Ferber, O. Gutknecht, and F. Michel. From Agents to Organizations: an Organizational View of Multi-Agent Systems. In P. Giorgini, J. Muller, and J. Odell, editors, *Agent-Oriented Software Engineering VI*, volume LNCS 2935 of *Lecture Notes in Computer Science*, pages 214–230. Springer-Verlag, 2004.
- [18] B. Gateau, O. Boissier, D. Khadraoui, and E. Dubois. Moiseinst: An organizational model for specifying rights and duties of autonomous agents. *Third European Workshop on Multi-Agent Systems (EUMAS 2005)*, pages 484–485, 2005.
- [19] B. Gateau, O. Boissier, D. Khadraoui, and E. Dubois. Moiseinst: An organizational model for specifying rights and duties of autonomous agents. *Environments for Multi-Agent Systems III*, 4389:41–50, 2007.
- [20] P. Giorgini, M. Kolp, and J. Mylopoulos. Multi-agent architectures as organizational structures. *Autonomous Agents and Multi-Agent Systems*, 13(1):3–25, 2006.
- [21] B. Horling. *Quantitative organizational modeling and design for multi-agent systems*. PhD thesis, 2006.
- [22] B. Horling and V. Lesser. Using odml to model multi-agent organizations. In *IAT '05: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 2005.
- [23] J. Hubner, J. Sichman, and O. Boissier. S-moise+: A middleware for developing organised multi-agent systems. In *In Proc.Int. Workshop on Organizations in Multi-Agent Systems, from Organizations to Organization Oriented Programming in MAS*, volume 3913 of *LNCS*, pages 64–78, 2006.
- [24] F. Lopez, M. Luck, and M. d’Inverno. A normative framework for agent-based systems. *Computational and Mathematical Organization Theory*, 12:227–250, 2006.
- [25] X. Mao and E. Yu. Organizational and social concepts in agent oriented software engineering. In *AOSE IV*, volume 3382 of *Lecture Notes in Artificial Intelligence*, pages 184–202, 2005.
- [26] J. Pavon, J. Gomez-Sanz, and R. Fuentes. *The INGENIAS Methodology and Tools*, volume chapter IX, page 236–276. Henderson-Sellers, 2005.
- [27] A. O. Software. Jack intelligent agents: Jack teams manual, release 4.1. 2004.
- [28] J. Sudeikat, L. Braubach, A. Pokahr, and W. Lamersdorf. Evaluation of agent-oriented software methodologies examination of the gap between modeling and platform. *AOSE-2004 at AAMAS04*, 2004.
- [29] I. Trencansky and R. Cervenka. Agent modelling

language (AML): A comprehensive approach to modelling mas. In *Informatica*, volume 29(4), pages 391–400, 2005.

- [30] J. Vazquez-Salceda, V. Dignum, and F. Dignum. Organizing multiagent systems. Technical report uu-cs-2004-015, Institute of Information and Computing Sciences, Utrecht University, 2006.
- [31] M. Wooldridge, N. R. Jennings, and D. Kinny. The Gaia Methodology for Agent-Oriented Analysis and Design. *Journal of Autonomous Agents and Multi-Agent Systems*, 15, 2000.