Programming Organization-Aware Agents
A Research Agenda

M. Birna van Riemsdijk Koen Hindriks Catholijn Jonker

Technische Universiteit Delft, The Netherlands
{m.b.vanriemsdijk,k.v.hindriks,c.m.jonker}@tudelft.nl

Abstract. Organizational notions such as roles, norms (e.g., obligations and permissions), and services are increasingly viewed as natural concepts to manage the complexity of software development. In particular in the context of multi-agent systems, agents are expected to be organization-aware, i.e., to understand and reason about the structure, work processes, and norms of the agent organization in which they operate. In this paper, we analyze which kinds of reasoning an agent should be able to do to function in an organization. We categorize these kinds of reasoning with respect to several dimensions, and distinguish three general approaches on how these might be integrated in existing agent programming languages. Through this, we provide a research agenda on what needs to be addressed when developing techniques for programming organization-aware agents.

1 Introduction

Software systems are becoming increasingly complex. One of the main challenges in the field of software engineering is to develop tools and techniques for managing this complexity [41]. A central role is played by development methodologies and programming languages, which can help managing complexity by providing appropriate *concepts and abstractions* in terms of which an application can be analyzed, designed, and implemented. Searching for the "most appropriate" (most convenient, most natural, most succinct, most efficient, most comprehensible, ...) programming concepts and abstractions is addressed in programming language design [5, Foreword].

In the field of multi-agent systems (MAS), several dedicated agent programming languages have been proposed [5] to support the implementation of MAS. The programming abstractions on which many of these languages focus, are aimed at programming how an agent can reach certain goals, how it should react to events occurring in its environment, and how it should communicate with other agents.

A line of research in the MAS field that has received increasing attention in the last years, is to assign an organization to the MAS with the aim of organizing and regulating it (see, e.g., [17, 4, 31, 3, 54]), in a similar way as done in human organizations. Using an *organizational specification* to organize a MAS should make the agents more effective in attaining their purpose, or prevent certain

undesired behavior from occurring. An organizational specification may define the structure of the agent organization in terms of roles and the relations between roles, and specify the norms (e.g., obligations and prohibitions) that are to be followed by the agents of the MAS.

Agents that operate in such an organized MAS are expected to *take the specification of the organization, as well as their own position in the organization, into account when deciding what to do*. For example, an agent playing the role of supervisor can typically delegate tasks to its subordinates, but not the other way around. Agents playing the role of supervisor or subordinate should be aware of this and take this into account when deciding on action, if they are to operate effectively in the organization. While there is a growing body of work addressing the modeling and implementation of organizational specifications, little research has been done on how to program agents that use these specifications for deciding on action (see [57, 11, 18, 12, 31, 40] for a few papers that do address various aspects of this). We call agents that are capable of such organizational reasoning and decision making *organization-aware agents*.

It is the aim of this paper to analyze which kinds of reasoning an agent should be able to do to function in an organization and how this might be programmed, thereby providing a starting point for research on how to program an agent's organizational reasoning.

2 Motivation and Background

In this section, we explain in more detail why we believe that it is important that agents are capable of organizational reasoning and decision making, and we provide more background on the subject.

One of the main application areas that we believe would benefit from organization-aware agents, is *simulation and training* of human organizations. Agent-based simulation is important for analyzing and training organizations, especially where learning is dangerous or where teaching is expensive [50, 48, 28, 14, 1, 42, 49]. The more realistic the behavior of the agents is, the better the training results will most likely be.

For example, since 9/11 a lot of research has been put into the improvement of crisis management. Agent technology plays an important role in creating computer simulations for analysis support and in developing training environments for this domain [44, 46, 26, 49, 28, 27]. Software agents form a natural programming metaphor: there is a relatively close correspondence between real-world crisis management organizations and collections of autonomous agents that interact in a dynamic environment, usually with some individual or collective purpose.

As an example of the kind of organizational reasoning that is required in crisis management, consider the following scenario:

> *An explosion has occurred in a chemical plant and hazardous chemicals have leaked into the area. To prevent further injuries, it is essential that the emergency response team secures the area by setting up road blocks.*

Normally, this is the task of policemen, but firefighters are the first to arrive on the scene. Should the firefighters set up the road blocks?

The firefighters operate in the context of a larger emergency response organization, for which operating procedures and role responsibilities are (partly) described in a disaster plan. The firefighters have to decide whether to take on a role that is normally played by other members of the organization (according to the disaster plan), or whether to give priority to their own goal of fighting the fire. This kind of organizational reasoning and decision making takes place frequently in organizations such as those for crisis management. Endowing agents with similar capabilities is thus highly relevant.

Other application areas where organizational reasoning can be useful, are human-agent teamwork and open systems. In human-agent teamwork, humans and agents work together to achieve joint goals. It is essential for the effective operation of a (human-agent) team to create a shared understanding between teammates [52, 39, 36]. This is facilitated by making agents understand how to function as part of an organization. Open systems allow agents to enter and leave the system as it operates [19, 2, 15]. Typical examples are e-institutions such as market places on the internet. An open system is organized by an organizational specification that is to be followed by the participating agents. Organizational reasoning and decision making facilitates the functioning of agents as part of an open system.

Existing research on *organizational modeling languages* supports the specification of organizations using the notion of "role" (see, e.g., [19, 21, 17, 58, 30, 4, 31, 54, 33]). In this way, an organizational specification abstracts from the individual agents that will eventually play the roles, comparable to a disaster plan in crisis management: a disaster plan describes the desired structure and functioning of the crisis management organization without specifying which individuals will play the roles of policeman and firefighter in case of a crisis. Some organizational modeling languages come with implementation frameworks [20, 31, 45] that, for example, allow agents to access and modify the state of the organization and enforce organizational constraints by applying sanctions in case of their violation.

Given an organizational specification, it is up to the agents to operationalize it by playing the roles of the organization. A few abstract models exist on how aspects of an organizational specification may influence agent behavior [16, 7, 38, 18, 8], but little is understood on how to operationalize and combine them on the level of agent programming. First steps towards this have been made [57, 11, 18, 12, 31, 40]. However, in order to program agents that take reasoned decisions on how to play their part in the organization, more advanced forms of organizational reasoning and decision making are needed as discussed in the rest of this paper.

3 Dimensions of Organizational Reasoning

In this section, we categorize forms of organizational reasoning along three main dimensions: the phases of organizational participation of the agent, the elements of organizational specifications that agent should understand, and the direction of organizational reasoning (top-down starting from an organizational specification, or bottom-up where the agents have to figure out amongst themselves how to organize). It is the aim of this section to provide a reasoned overview of kinds of organizational reasoning. We have included references that can be used as a starting point if a certain kind of organizational reasoning is studied, and we provide several examples of organizational reasoning for each of the categories, such as deciding whether to take on a role (weighing possibly conflicting interests as in the firefighter scenario), reasoning about how to fulfill norms imposed by the organization and deciding when to go against them (even though sanctions might be applied), reasoning about how to change the organization, etc.

3.1 Phases of Organizational Participation

Organizations are operationalized through the agents that play roles in the organization. In the case of open organizations where agents may enter and leave the organization, it seems evident to distinguish three phases of an agent participating in an organization: entering the organization, playing roles in the organization, and leaving the organization. If closed systems are considered, only the second phase is relevant. In each phase, different kinds of reasoning can be distinguished.

Entering the Organization In this phase, the agent has to reason about whether it *wants* to enter the organization, and whether it has the *capabilities* to behave as the organization requires. That is, the agent needs to consider what it wants from the organization, and what the organization wants from it.[1] Since agents typically participate in an organization by playing a role in the organization, this kind of reasoning will focus on deciding whether to play a role.

In order to determine this, the agent has to reason about whether playing a certain role in the organization can help it to fulfill its own goals, and whether it can come to an agreement, e.g., with respect to the interaction protocols that will be used. For this, the agent has to *understand the specification of the role*, and should be able to relate it to its own *goals*. For example, if an agents wants to sell books, it might participate as a seller in an auction if it understands that playing the seller role will enable it to sell the books. The agent also has to determine whether it can play the role in the way required from the organization. For example, the auction might require the agent to have a bank account in the

[1] See also [8], where the notion of social power is used as a basis for the agent to decide whether it wants to enter a group, i.e., whether the power it loses by entering is compensated by the power it gains or not.

country where the auction is held, to enable easy transfer of the money earned by selling the books. In order to determine whether the agent can fulfill these norms belonging to a role, the agent has to understand them and relate them to its own capabilities. For example, if the agent currently does not have a bank account in the country where the auction is held, but it does know how to obtain such a bank account, it might decide to play the role and open up the account.

In practice, it will often be the case that the agent and the role do not match exactly [11]. Playing the role might not enable the agent to fulfill its goals entirely, and the agent might not have all the capabilities in principle required to play the role. The agent can then *negotiate* about the terms under which it plays the role in the organization (see also [36], where goal negotiation is identified as one of the challenges of human-agent cooperation), and form a contract [17] with the organization. A related aspect is that *conflicts* may arise between some goals of the agent and requirements imposed by the role [11] (see also [53, 47, 56] for work on conflicting goals). For example, the agent may want to sell its books to only one buyer because it concerns a book series that the agent feels should not be separated. However, the auction does not offer the possibility to specify that the books should be sold to a single buyer. The agent will then have to reason about *priorities between its goals*, i.e., whether selling the books is more important than keeping the books together.

Playing Roles in the Organization When the agent has decided to play a role in the organization, it has to decide how to do this. In other words, the abstract specification of the role with accompanying norms and responsibilities or goals has to be interpreted and translated to concrete actions taken by the agent. We distinguish three increasingly advanced levels of organizational reasoning related to playing roles: behaving according to the specification of the role, reasoning about violation of the specification, and reasoning about adapting the specification of the role (or other parts of the organization).

Before we discuss these levels of organizational reasoning, we remark that an agent may also be *designed* to fulfill certain roles in an organization (see, e.g., the Gaia methodology [60])[2]. An agent may then always behave according to the specification of the role by design, and no advanced forms of reasoning are required to make sure the agent complies with the specification. Generation of agent skeleton code [57] from the organizational specification can be used to ease agent developed in such a setting.[3] While this is useful for certain kinds of applications, it limits the flexibility of the MAS. We aim for domains in which increased flexibility obtained through organizational reasoning would provide additional benefits. For example, in the case of open systems it would be beneficial if the agents are capable of entering and functioning in various organizations

[2] In [34], the ROADMAP methodology is proposed which extends Gaia, among other things by allowing agents to change roles at run-time.

[3] Although the skeletons make sure the agent behaves according to the specification, even in this approach reasoning may still be required to choose from several allowed options.

without the programmer having to specify for each of these organizations exactly how the agent should do this. Also, in simulation of organizations one may want to simulate with different norms, organizational structures, and role specifications, in order to try out what yields better results. If the agents are able to understand the organizational specification, this provides for additional flexibility and a kind of separation of concerns: one could modify the organizational specification independently of the agents, and the agents would be able to adapt to this. The discussion in the sequel focuses on organizational reasoning in this sense.

The first level of organizational reasoning that we distinguish, *behaving according to the specification*, requires that the agent understands the role specification and is able to translate it into concrete actions that fulfill the goals of the role and do not violate the norms of the role. One of the main challenges here is to bridge the gap between the possibly abstract specification of the role, and concrete actions that the agent has to take (see also [59] for an approach to concretizing norms for electronic institutions). For example, the role specification of a policeman might specify that he should keep people away from hazardous situations. The agent playing the role of policeman then has to translate this into setting up road blocks in case hazardous chemicals have escaped from a plant. The closer the role specification is to the internal agent architecture (e.g., in [11, 12] they are relatively close), the easier this translation will presumably be. If a role is specified in an organizational modeling language and the agent is programmed in an agent programming language that was not designed to work with the organizational modeling language, bridging this gap will be more difficult. Another challenge is to let the agent reason about its own behavior to prevent it from violating norms (see, e.g., [24, 25] for work on how to prevent violation of goals, and [40] for an approach on how AgentSpeak(L) agents can adapt their behavior to norms). For example, if the norm is that policemen should always execute tasks in pairs, this should influence the agents' actions accordingly.

The second level, *reasoning about violation*, requires the agent to decide whether, even though it has decided to play the role, it will nevertheless violate some of the requirements imposed by the role (see also [10, 7]). For example, the role of policeman might specify that it can only be played by agents that have the corresponding diploma. However, if road blocks need to be set up and policemen have not arrived yet, firefighters might decide to play the role of policemen and set up the road blocks, even though strictly speaking they are not allowed to do this. Deciding on whether to violate the role specification requires *weighing* the benefits of breaking the rules against possible negative consequences or sanctions resulting from this. Here, it should be noted that there is a difference between norm *enforcement*, which makes the violation of norms less desirable by introducing, e.g., sanctioning mechanisms, and norm *regimentation*, in which case it is made impossible to violate a norm (see, e.g., [35, 55]). In the former case, an agent can decide to violate the norm and accept the sanction, while in the latter case this is not possible. Regimentation can be realized using organizational middleware in which the agent sends requests for actions that it would

like to execute to the middleware, which decides whether the specific action is indeed executed. For example, in MOISE$^+$ [31] each agent is connected to a so-called "Orgbox" which forms the interface between agents and middleware, and in ISLANDER/AMELI [20] governors are used for mediating the participation of agents in an electronic institution.

The third level, *reasoning about adaptation*, requires the agent to reason about how possible changes to the specification of the role or organization as a whole might affect the functioning of the organization (see [30] for an approach that uses reorganization agents for performing reorganizations). In particular, the agent should be able to determine which changes will lead to improvements in the functioning of the organization. This might, for example, be determined by comparing the actual behavior of the organization against the prescribed behavior. Discrepancies can indicate that changes are necessary.

Besides reasoning about how to play a role in the organization, the agent should also reason about whether to take on additional roles in the organization or change roles. This is largely similar to the kind of reasoning needed when entering the organization. An additional aspect that needs to be addressed if the agent decides to take on multiple roles, is that it should reason about possible conflicts between the requirements imposed by these roles. For example, if an incident occurs, the mayor of the city might take on the role of coordinator of the rescue efforts. This role requires him to be at the crisis management centre. However, this conflicts with his role as mayor, as in that role he should be at the scene to comfort the injured people.

Leaving the Organization In order to enter this phase, the agent has to reason about whether it still wants to participate in the organization. Reasons for leaving the organization (see also [29]) can be that the agent has achieved the goals it wanted to by being part of the organization. Another reason can be that it believes it will not be able to achieve its goals. These considerations are similar to an agent deciding whether to drop a goal in BDI agent languages: when the goal is achieved or believed to be unachievable. Another reason for the agent to leave the organization is that it is thrown out, for example because it has not adhered to the norms of the organization. Alternatively, an agent may not be allowed to leave the organization, e.g., if it has not payed yet in the case of an electronic market place.

3.2 Elements of Organizational Specifications

As explained in Section 1, an organizational specification defined in an organizational modeling language is used to organize and regulate a multi-agent system. For example, the MOISE$^+$ organizational modeling language [4, 31] specifies an organization in terms of a structural dimension using the notions of roles and groups, a functional dimension that describes how global collective goals should be achieved, and a normative dimension expressing permissions and obligations for roles, related to the achievement of (sub)goals.

Agents capable of organizational reasoning should be able to understand and reason about an organizational specification. This requires agents to understand all *elements* of the organizational specification, such as the structural dimension, the functional dimension and the normative dimension in the case of MOISE$^+$. Different kinds of organizational reasoning are associated with each of these elements.

For example, the MOISE$^+$ structural dimension is related to the communication between agents. It specifies that agents playing a certain role should not communicate with agents playing some other role. Moreover, the structural dimension allows the specification of authority links between roles. This should presumably influence how agents handle and pose requests: if a higher ranked agent requests something from a lower ranked agent, the latter should usually obey (unless its reasoning determines that there are strong arguments against obeying).

3.3 Direction of Organizational Reasoning

The third dimension that we distinguish, is the direction of organizational reasoning: top down or bottom up. By top down reasoning we mean that the agents take an organizational specification as the basis for their behavior in the organization, for example by following the specified work processes. By bottom up reasoning we mean that the agents figure out amongst themselves how they should cooperate, without a predefined organization. Naturally, these can co-exist. In general, the more is specified in the organizational specification, the less room there will be for bottom-up reasoning, and vice versa. This dimension is related to the distinction made between agent-centered and system-centered [31], where the latter refers to the case where there is an explicit organizational specification. The direction of organizational reasoning refers to the kind of reasoning required from the agents in each of these cases.

In the discussion so far, we have focused mainly on top down reasoning. In bottom up reasoning, an important kind of reasoning is *reasoning about other agents*. This has received relatively little attention in the agent programming literature. Nevertheless, we mention several approaches in agent programming and other areas of MAS research that address aspects of reasoning about other agents. For example, for virtual characters in games, it is important that they are able to form an internal representation of the mental states of other players, e.g., for predicting what they might do [37, 51]. A related area is plan or intention recognition [9, 13, 23]. There, agents try to recognize the plans or intentions of other agents (which might also be humans), based on observations of their actions and domain knowledge of certain standard procedures. Reasoning about humans is also an important part of mixed-initiative systems, in which the system collaborates with a user by sometimes taking initiative to support the user's activities (see, e.g., [43, 22]). It will have to be investigated whether techniques developed in these areas can be used in the context of organized MAS for reasoning about other agents, and how this kind of bottom up reasoning can be combined with top down reasoning.

4 Programming Organization-Aware Agents

In Section 3, we have categorized kinds of organizational reasoning. In this section, we discuss general approaches for how agents could be programmed to perform organizational reasoning. Future research will have to make an effort in clearly identifying their respective strengths and weaknesses.

The first approach that we identify, is to *use existing agent programming languages* for specifying an agent's organizational reasoning. This is the approach taken in [31], where Jason is used to program agents that should function in a MOISE$^+$ organization. Jason is extended by necessary organizational actions, such as an action for adopting a role. Otherwise, no additions to Jason are made. The actions do not come with sophisticated reasoning, e.g., the action for adopting a role does not check whether the agent has the capabilities for playing the role. Plans are programmed, e.g., to specify when the agent should adopt a particular role.

The advantage of this approach is that an existing language is used, which means that a programmer who knows Jason could in principle program Jason agents that should function in an organization. The disadvantage is that the programmer is not explicitly supported in programming organization-aware agents. For example, he will have to program how an agent should determine whether it can play a role in an organization. This is a non-trivial task. Also, some aspects of organizational reasoning have to be repeated for each instance of it. For example, an agent should notify the organization if it has achieved some goal. This means that an action has to be included in each plan where a goal of the organization is reached.

The second approach we discuss here aims for a more generic approach based on the recognition that in open systems software agents can no longer be completely hard-coded as this would require the reprogramming of agents every time a virtual organization changes. The idea is that agents instead need to exchange information about virtual organizations to be able to adequately coordinate their activities. Such information exchange needs to be supported by explicitly representing the structure and norms of the organization in some declarative language. Current state of the art agent programming languages provide an agent with the capability to reason with its beliefs and goals. In order to become organization-aware the second approach would be to suggest adding dedicated capabilities to an agent that support organizational reasoning. The basic idea here would be to add an "organizational attitude" to agents besides their epistemic and motivational attitude. This can be achieved by adding dedicated organizational reasoning patterns as *software components* or plugins that will provide agents with the capability to reason about generic issues related to the organization they participate in. For example, plugins might be provided (i) for identifying the benefits of taking part in the organization, (ii) for negotiating about interaction protocols, (iii) for monitoring of norm compliance, and possibly other typical reasoning patterns.

Finally, the third - even more ambitious - approach would be to develop *new programming abstractions* for programming an agent's organizational reasoning

and decision making. In this approach, the organizational reasoning would be done in a separate layer using the new programming abstractions, and the agent's cognitive reasoning and decision making (reasoning about achieving goals) in another. The latter is then programmed in existing agent programming languages. A semantic connection between both layers would have to be established. For example, organizational reasoning and decision making involves deciding whether to take on a certain role, which typically comes with permissions and obligations to achieve goals. If the agent takes on the role, this should influence its cognitive reasoning and decision making on how to reach the accompanying goals. However, if it turns out that the agent is not able to achieve the goals after all, or decides to achieve other goals because this is more in line with its own interests, this should in turn influence its organizational reasoning and decision making. It may, e.g., have to decide to delegate some goals to other agents (see also [32, 6]). An architecture for this approach is depicted in Figure 1.

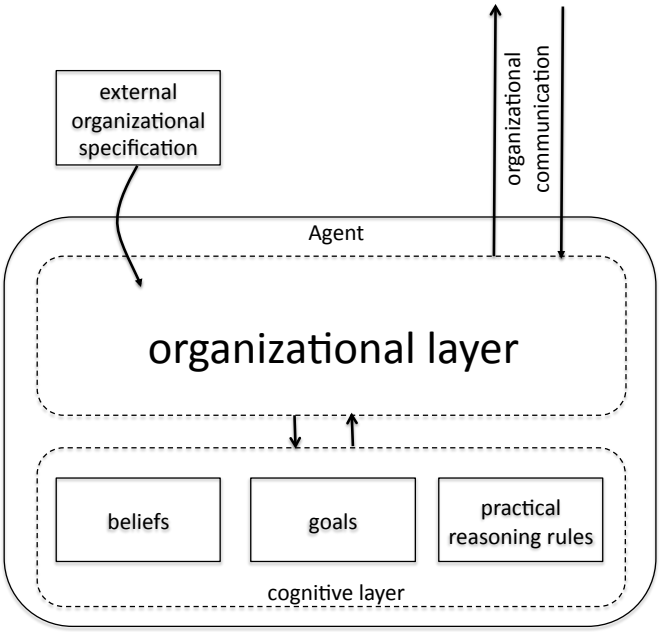

Fig. 1. Architecture for Organizational Reasoning

Programming languages for organization-aware agents would reduce the effort needed for programming and maintaining agents. We refer to the well-known rule of thumb that programmers can code a fixed number of lines of code per

hour, independently of the language in which the coding is done. Having programming constructs for organization-aware agents would allow the programmer to represent the reasoning in a concise way, reducing the amount of time needed for programming. Moreover, it would provide conceptual support, since the language would guide the programmer in its thinking about the problem. For maintenance of programs, programs written in terms of higher programming abstractions are easier to read and thus to maintain than programs written in lower-level languages. Maintenance of software is notoriously hard for those that did not write the programs themselves, and increased understandability of the code would improve maintenance.

Although we aim for applications in which an agent's ability to perform organizational reasoning is beneficial, it may not be necessary for the agent to perform all of the reasoning required to function in an organization itself. Some approaches alleviate the agent partly from having to do organizational reasoning, by letting the organizational middleware take over some aspects of this. For example, in ISLANDER/AMELI, governors can be posed questions about the institution [19], in J-MOISE+ events are sent to agents to notify them, e.g., about whether the agent has an obligation, and in [55] the agent can delegate tasks to its role, which forms the connection between the agent and the organization. Although these approaches can help explain the agent what is expected from it in a certain situation from the perspective of the organization, the agent will still have to take the decision as to what it will do. That is, either decide which of several allowed actions it takes in case of regimentation (see, e.g., [57]), or decide whether to comply with a norm if the norm is only enforced.

5 Conclusion

In this paper, we have identified kinds of organizational reasoning along three dimensions: phases of organizational participation, elements of organizational specifications, and direction of organizational reasoning. Moreover, we have identified three approaches for programming organization-aware agents: using existing agent programming languages, developing components for organizational reasoning, and developing dedicated programming abstractions for supporting organizational reasoning. Through this, we have provided a research agenda on what needs to be addressed when developing techniques for programming organization-aware agents.

Acknowledgements

We would like to thank the anonymous referees for their useful comments.

References

1. AOS group. Jack: an agent infrastructure for providing the decision-making capability for autonomous systems (whitepaper). `http://www.aosgrp.com/downloads/JACK\_WhitePaper\_UKAUS.pdf`.

2. J. L. Arcos, M. Esteva, P. Noriega, J. A. Rodríguez-Aguilar, and C. Sierra. Engineering open environments with electronic institutions. *Engineering applications of artificial intelligence.*, 18(2):191–204, 2005.
3. M. Baldoni, G. Boella, V. Genovese, R. Grenna, and L. van der Torre. How to program organizations and roles in the JADE framework. In *6th German Conference on Multiagent System Technologies (MATES'08)*, volume 5244 of *LNCS*, pages 25–36. Springer, 2008.
4. O. Boissier, J. F. Hübner, and J. S. Sichman. Organization oriented programming: From closed to open organizations. In *Proceedings of the 7th International Workshop on Engineering Societies in the Agents World (ESAW'07)*, volume 4457 of *LNCS*, pages 86–105. Springer, 2007.
5. R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni. *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.
6. F. Brazier, C. Jonker, and J. Treur. Formalization of a cooperation model based on joint intentions. In J. Mueller, M. Wooldridge, and N. Jennings, editors, *Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages (ATAL'96)*, volume 1193 of *LNAI*, pages 141–155. Springer-Verlag, 1997.
7. J. Broersen, M. Dastani, J. Hulstijn, and L. van der Torre. Goal generation in the BOID architecture. *Cognitive Science Quarterly*, 2(3-4):428–447, 2002.
8. C. Carabelea, O. Boissier, and C. Castelfranchi. Using social power to enable agents to reason about being part of a group. In *Proceedings of 5th International Workshop on Engineering Societies in the Agents World (ESAW'04)*, volume 3451 of *LNCS*, pages 166–177. Springer, 2005.
9. S. Carberry. Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11:31–48, 2001.
10. C. Castelfranchi, F. Dignum, C. Jonker, and J. Treur. Deliberative normative agents: Principles and architecture. In *6th International Workshop on Intelligent Agents VI, Agent Theories, Architectures, and Languages (ATAL'99)*, volume 1757 of *LNCS*, pages 364–378. Springer, 2000.
11. M. Dastani, V. Dignum, and F. Dignum. Role-assignment in open agent societies. In *Proceedings of the second international joint conference on autonomous agents and multiagent systems (AAMAS'03)*, pages 489–496, Melbourne, 2003.
12. M. Dastani, M. B. van Riemsdijk, J. Hulstijn, F. Dignum, and J.-J. Ch. Meyer. Enacting and deacting roles in agent programming. In J. Odell, P. Giorgini, and J. Müller, editors, *Agent-Oriented Software Engineering V*, volume 3382 of *LNCS*, pages 189–204. Springer-Verlag, 2005.
13. R. Demolombe and A. M. O. Fernandez. Intention recognition in the situation calculus and probability theory frameworks. In *6th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA'05)*, volume 3900 of *LNCS*, pages 358–372. Springer, 2006.
14. F. Dignum, V. Dignum, and C. Jonker. Towards agents for policy making. In *Proceedings of the 9th International Workshop on Multi-Agent-Based Simulation (MABS'08)*, 2008.
15. F. Dignum, V. Dignum, J. Thangarajah, L. Padgham, and M. Winikoff. Open agent systems??? In *Proceedings of the 8th International Workshop on Agent-Oriented Software Engineering (AOSE'07)*, volume 4951 of *LNCS*, pages 73–87. Springer, 2008.
16. F. Dignum, D. Kinny, and L. Sonenberg. From desires, obligations and norms to goals. *Cognitive Science Quarterly*, 2(3-4):407–430, 2002.

17. V. Dignum. *A Model for Organizational Interaction: Based on Agents, Founded in Logic*. PhD thesis, 2004.
18. V. Dignum and F. Dignum. What's in it for me? Agent deliberation on taking up social roles. In *Proceedings of the second European Workshop on Multi-Agent Systems (EUMAS'04)*, 2004.
19. M. Esteva, J. Padget, and C. Sierra. Formalizing a language for institutions and norms. volume 2333 of *LNCS*, pages 348–366. Springer, 2002.
20. M. Esteva, B. Rosell, J. A. Rodríguez-Aguilar, and J. L. Arcos. AMELI: An agent-based middleware for electronic institutions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'04)*, pages 236–243. IEEE Computer Society, 2004.
21. J. Ferber, O. Gutknecht, and F. Michel. From agents to organizations: An organizational view of multi-agent systems. In *Proceedings of 4th International Workshop on Agent-Oriented Software Engineering (AOSE'03)*, volume 2935 of *LNCS*, pages 214–230. Springer, 2003.
22. G. Ferguson and J. Allen. Mixed-initiative systems for collaborative problem solving. *AI Magazine*, 28(2), 2009.
23. A. Goultiaeva and Y. Lespérance. Incremental plan recognition in an agent programming framework. In *Workshop on Plan, Activity, and Intent Recognition (PAIR'07)*, 2007.
24. K. Hindriks and M. B. van Riemsdijk. Satisfying maintenance goals. In *Declarative Agent Languages and Technologies V (DALT'07)*, volume 4897 of *LNAI*, pages 86–103. Springer, 2008.
25. K. Hindriks and M. B. van Riemsdijk. Using temporal logic to integrate goals and qualitative preferences into agent programming. In *Declarative Agent Languages and Technologies VI (DALT'08)*, volume 5397 of *LNAI*, pages 215–232. Springer, 2009.
26. M. Hoogendoorn, C. Jonker, V. Popova, A. Sharpaskykh, and L. Xu. Formal modelling and comparing of disaster plans. In *Proceedings of the Second International Conference on Information Systems for Crisis Response and Management (ISCRAM'05)*, pages 97–107, 2005.
27. M. Hoogendoorn, C. Jonker, P. van Maanen, and A. Sharpanskykh. Formal analysis of empirical traces in incident management. *Reliability Engineering and System Safety*, 93:1422–1433, 2008.
28. M. Hoogendoorn and J. Treur. An adaptive multi-agent organization model based on dynamic role allocation. In *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'06)*, pages 474–481. IEEE Computer Society, 2006.
29. N. Hormazábal, H. L. Cardoso, J. L. de la Rosa, and E. Oliveira. An approach for virtual organizations' dissolution. In *Proceedings of the international workshop on coordination, organization, institutions and norms in agent systems (COIN'09@AAMAS)*, pages 93–108, 2009.
30. J. F. Hübner, J. S. Sichman, and O. Boissier. Using MOISE+ for a cooperative framework of MAS reorganisation. In *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence (SBIA'04)*, volume 3171 of *LNAI*, pages 506–515. Springer, 2004.
31. 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*, 1(3/4):370–395, 2007.

32. N. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence Journal*, 74(2), 1995.
33. C. Jonker and J. Treur. From organisational structure to organisational behaviour formalisation. *International Journal of Agent-Oriented Software Engineering*, 2009. To appear.
34. T. Juan, A. R. Pearce, and L. Sterling. ROADMAP: extending the Gaia methodology for complex open systems. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, pages 3–10. ACM, 2002.
35. L. Kamara, J. Pitt, and M. Sergot. Norm-aware agents for ad hoc networks: A position paper. In *Proceedings of the AAMAS'04 Workshop on Agents and Ubiquitous Computing*, 2004.
36. G. Klein, D. D. Woods, J. M. Bradshaw, R. R. Hoffman, and P. J. Feltovich. Ten challenges for making automation a "team player" in joint human-agent activity. *IEEE Intelligent Systems*, 19(6):91–95, 2004.
37. J. E. Laird. It knows what you're going to do: adding anticipation to a quakebot. In *Proceedings of the fifth international conference on Autonomous Agents*, pages 385–392. ACM, 2001.
38. F. López y López. *Social Power and Norms: Impact on Agent Behaviour*. PhD thesis, 2003.
39. E. Mathieu, T. S. Heffner, G. Goodwin, E. Salas, and J. Cannon-Bowers. The influence of shared mental models on team process and performance. *The Journal of Applied Psychology*, 85(2):273–283, 2000.
40. F. Meneguzzi and M. Luck. Norm-based behaviour modification in BDI agents. In *Proceedings of the eighth international joint conference on autonomous agents and multiagent systems (AAMAS'09)*, pages 177–184, Budapest, 2009.
41. R. J. Mitchell, editor. *Managing Complexity in Software Engineering*. Institution of Electrical Engineers, UK, 1990.
42. S. Munroe, T. Miller, R. A. Belecheanu, M. Pechoucek, P. McBurney, and M. Luck. Crossing the agent technology chasm: Experiences and challenges in commercial applications of agents. *Knowledge Engineering Review*, 21(4):345–392, 2006.
43. K. Myers and N. Yorke-Smith. Proactivity in an intentionally helpful personal assistive agent. In *Proceedings of AAAI 2007 Spring Symposium on Intentions in Intelligent Systems*, 2007.
44. R. Nair, M. Tambe, and S. Marsella. Team formation for reformation in multiagent domains like RoboCupRescue. In *International Symposium on RoboCup (RoboCup'02)*, 2002.
45. D. Okouya and V. Dignum. OperettA: a prototype tool for the design, analysis and development of multi-agent organizations. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems (AAMAS'08)*, pages 1677–1678, Richland, SC, 2008. International Foundation for Autonomous Agents and Multiagent Systems.
46. A. Oomes. Organization awareness in crisis management: dynamic organigrams for more effective disaster response. In *Proceedings of the First International Conference on Information Systems for Crisis Response and Management (ISCRAM'04)*, pages 63–68, 2004.
47. A. Pokahr, L. Braubach, and W. Lamersdorf. A goal deliberation strategy for BDI agent systems. In *MATES 2005*, volume 3550 of *LNAI*, pages 82–93. Springer-Verlag, 2005.
48. W. B. Rouse and K. R. Boff. *Organizational Simulation*. Wiley, 2005.

49. N. Schurr, P. Patil, F. Pighin, and M. Tambe. Using multiagent teams to improve the training of incident commanders. In *Proceedings of the fifth international joint conference on autonomous agents and multiagent systems (AAMAS'06), Industry Track*, Hakodate, 2006.
50. M. Sierhuis. *Modeling and Simulating Work Practice; Brahms: A multiagent modeling and simulation language for work system analysis and design*. PhD thesis, 2001.
51. M. Sindlar, M. Dastani, F. Dignum, and J.-J. Ch. Meyer. Mental state abduction of bdi-based agents. In *Declarative Agent Languages and Technologies VI (DALT'08)*, volume 5397 of *LNAI*, pages 161–178. Springer, 2009.
52. K. Sycara and G. Sukthankar. Literature review of teamwork models. Technical Report CMU-RI-TR-06-50, Carnegie Mellon University, 2006.
53. J. Thangarajah, L. Padgham, and M. Winikoff. Detecting and avoiding interference between goals in intelligent agents. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003)*, 2003.
54. N. A. Tinnemeier, M. Dastani, and J.-J. Ch. Meyer. Orwell's nightmare for agents? Programming multi-agent organisations. In *Proceedings of the Fifth International Workshop on Programming Multiagent Systems (ProMAS'08)*, 2008.
55. N. A. Tinnemeier, M. Dastani, and J.-J. Ch. Meyer. Roles and norms for programming agent organizations. In *Proceedings of the eighth international joint conference on autonomous agents and multiagent systems (AAMAS'09)*, pages 121–128, Budapest, 2009.
56. M. B. van Riemsdijk, M. Dastani, and J.-J. Ch. Meyer. Goals in conflict: Semantic foundations of goals in agent programming. *Autonomous Agents and Multi-Agent Systems*, 18(3):471–500, 2009.
57. W. W. Vasconcelos, J. Sabater, C. Sierra, and J. Querol. Skeleton-based agent development for electronic institutions. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02)*, pages 696–703. ACM, 2002.
58. J. Vazquez-Salceda. *The Role of Norms and Electronic Institutions in Multi-Agent Systems: The HARMONIA Framework*. Whitestein Series in Software Agent Technologies and Autonomic Computing. Birkhäuser, 2004.
59. J. Vázquez-Salceda, H. Aldewereld, D. Grossi, and F. Dignum. From human regulations to regulated software agents' behavior. *Journal of Artificial Intelligence and Law*, 16(1):73–87, 2008.
60. F. Zambonelli, N. R. Jennings, and M. Wooldridge. Developing multiagent systems: The Gaia methodology. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 12(3):317–370, 2003.