Dimensions of Organizational Coordination

Jensen, Andreas Schmidt; Aldewereld, Huib; Dignum, Virginia

Published in:
Proceedings of the 25th Benelux Conference on Articial Intelligence

Publication date:
2013

Document Version
Publisher's PDF, also known as Version of record

Dimensions of Organizational Coordination

Andreas Schmidt Jensen [a] Huib Aldewereld [b] Virginia Dignum [b]

[a] *Technical University of Denmark, Kgs. Lyngby, Denmark*
[b] *Delft University of Technology, Delft, The Netherlands*

Abstract

It is hard, if not impossible, to assume anything about agents' behavior in a society with heterogeneous agents from different sources. Organizations are used to restrict and guide the agents' actions such that the global objectives of the society are achieved. We discuss how agents can be supported to include organizational objectives and constraints into their reasoning processes by considering two alternatives: agent reasoning and middleware regulation. We show how agents can use an organizational specification to achieve organizational objectives by delegating and coordinating their activities with other agents in the society, using the GOAL agent programming language and the OperA organizational model.

1 Introduction

Organizations can be defined as a set of entities regulated by mechanisms of social order and created by more or less autonomous actors to achieve common goals. Agents can enact roles, or take specific positions, in a society and as such interact with others as a means to accomplish their own goals [4]. In the simplest case, a society is an environment that is accessible by anyone and that has no control over the agents entering it. Without any control, it is not possible for the society to ensure that global objectives are achieved and it is furthermore very hard to assume any kind of behavior of the agents in the system. Consider a hospital: if anyone can enter and exit the hospital whenever they want, can go anywhere they want inside the hospital, and there is no restriction on who can enact the role as a doctor or nurse, it is very hard to ensure that sick people will get the treatment they need. Clearly, the society needs some kind of mechanism to ensure that certain objectives are achieved and that predefined boundaries are not violated. In multi-agent systems, an organization is the usual mechanism used for controlling the agents entering the society. An organization is usually defined via roles, groups, objectives and norms, and it provides a more or less abstract description of what is expected of the agents in the society. The organizational specification should be balanced as to not decrease the agents' autonomy too much, since we after all still want the agents to exhibit an intelligent behavior. If the objectives are specified very generally, the agents are free to achieve them in a way that suits them best, whether this is by doing everything themselves, cooperating with other agents having the same (or similar) tasks, or even delegating some tasks to other agents. On the other hand, if objectives are specified in much detail by sub-objectives, there is a greater chance that they are achieved in the intended way. For example, a surgical procedure needs to follow some strict guidelines, but there should still be space for the surgeon to be able to adapt to different situations and act accordingly, if something unexpected happens.

In this paper, we investigate the mechanisms required for agents to behave intelligently in a system specified by an organizational model. We therefore take the top-down view of organizations and agents, namely that an organization is first specified and the agents are expected to adhere to that specification, rather than the bottom-up view, which is that the organization emerges as agents cooperate and achieve their goals. Our focus is on how to ensure that the agents enacting roles in an organization will eventually achieve

their (role) objectives, while still giving them enough freedom to decide by themselves how to do so. That is, even though our view is top-down, we investigate what is required for *the agents* to reason about their organizational objectives and what implications their role relations have on it. We begin with an analysis of the requirements of organizational reasoning, eventually leading to the main contribution of the paper: building blocks that enables agents to perform organizational reasoning about completion, coordination and delegation of tasks.

We base our work on the OperA model in which roles amongst other things are specified by their objectives, which in OperA can be part of a so-called landmark patterns that provides the order in which the objectives should be completed. We aim to show that while this ordering may take away some of the agents' freedom (things have to be done in a specific order), the agents can still remain autonomous as long as the ordering is not too fine-grained. We further elaborate on how the agents can choose to achieve their objectives differently based on the situation and the organizational specification, such that they are able to e.g. delegate responsibilities to more capable agents or cooperate with agents that have the same objectives. We use the GOAL agent programming language to illustrate some of the code patterns that can be used for the organizational reasoning.

The rest of the paper is organized as follows. We begin with a brief introduction of the OperA model in section 2. In section 3, we present the approach we are taking in this paper. In section 4, we show how agents can cooperatively achieve their objectives using landmarks. We discuss related work and conclude the paper by discussing future research directions in section 5.

2 Organization modeling: The OperA Model

This section introduces the parts of the OperA model required for our work and describes a scenario that will be used as an example throughout the paper.

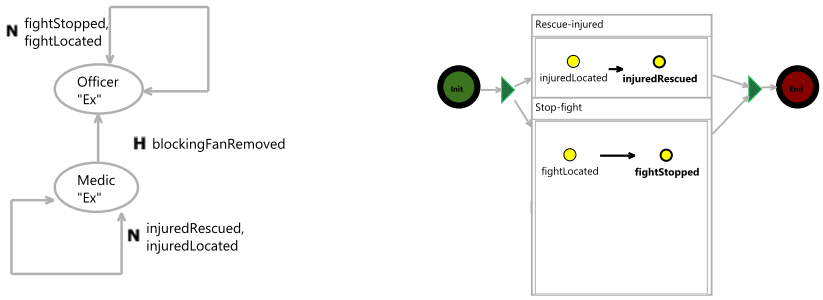
The OperA model [1] proposes an expressive way for defining open organizations distinguishing explicitly between the organizational aims, and the agents who act in it. That is, OperA enables the specification of the organizational structure, requirements and objectives, and at the same time allows participants to have the freedom to act according to their own capabilities and demands. At an abstract level, an OperA model describes the aims and concerns of the organization with respect to the social system. These are described as the organization's externally observable objectives, i.e. the desired states of affairs for the organization.

The OperA model contains the *Organizational Model* (OM), which is the result of the observation and analysis of the domain and describes the desired behavior of the organization, as determined by the organizational stakeholders in terms of objectives, norms, roles, interactions and ontologies. The OM consists of four interrelated *structures*: the social, interaction, normative and communicative structure. The *social structure* of an organization describes the roles holding in the organization. It consists of a list of role definitions, group definitions, and a role dependencies graph. The *interaction structure* describes the states that the agents should achieve, in terms of meaningful scenes that follow pre-defined abstract scene scripts. A scene script describes a scene by its players (roles), its desired results and the norms regulating the interaction. A scene script establishes also the desired interaction patterns between roles, that is, a desired combination of the (sub-) objectives of the roles. The *normative structure* describes expectations and boundaries for agent behavior, and the *communicative structure* specifies the ontology and the communication language used in the society.

In this paper, we focus on the social and interaction structures, specifically on roles, objectives and landmarks. We first describe the scenario we will use throughout the paper and then the relevant components of the OM will be described using the scenario.

2.1 Scenario: First Responders

We consider a scenario of first responders at a fight between groups of people, some of them being injured and requiring medical attention.



(a) Social Structure (b) Interaction Structure

Figure 1: Organizational specification for the scenario.

> After a match between Feyenoord and Ajax, groups of fans are fighting and some of the fans are badly hurt. The authorities have been contacted, and a group number of medics and police officers (the first-responders) have arrived. The medics are supposed to help the injured, while the police officers are supposed to break up the fight. However, fans of one group will not allow medics to help injured from the other group.

Several organizations exist in this scenario (each fan group, the medics, the police officers, and the first responders which contains both the medics and the police officers), but our focus is on the first responders. The overall goals of the first responders are to break up the fight and help any injured persons. Agents in the first responders organization are enacting either the role of a police officer or a medic. Police officers are able to break up fights and force people to move away from an area. Medics can assess whether a person is injured and they can move them to the ambulance. Medics cannot move injured people away from the scene, if fans are standing in their way, so they need to use force to move them away. Given the medics' abilities, this might not be possible, so in such situation they depend on a police officer helping them. Note that even though the scenario is quite simple, it touches upon some interesting and relevant dimensions of organizations, namely the clear purpose of the roles of the organization and the necessity of cooperation and delegation to achieve the organizational objectives effectively and successfully.

The organization can guide the agents in completing their objectives by giving them more or less freedom to decide how their objectives should be completed. Following the ideas of [8], we call such expressions *landmarks*, defined as conjunctions of logical expressions that are true in a state. Landmarks combined with a partial ordering to indicate the order in which the landmarks should be achieved are called a *landmark pattern*. For example, a fight must be located before it can be stopped.

The first responders organization, modeled in the OperettA environment [1], is shown in figure 1. Figure 1a shows the social structure of the OM, consisting of the two roles and their relations. Police officers depend on each other for locating and stopping the fight, while medics work together to locate and rescue injured fans. Furthermore, medics depend on police officers to remove fans that are blocking them. Figure 1b shows the relation between the scenes and the landmark patterns for the individual scenes. Note that several scenes can be happening at the same time and one agent can participate in different scenes simultaneously (e.g., a police officer is simultaneously participating in the *stop fight* and *rescue injured* scenes). Transitions also describe the conditions for the creation of a new instance of the scene. Note that we assume that role assignment has already been done, and figure 1b only show the relevant parts of the scene structure.

3 Dimensions of organizational reasoning

In open environments agents and organizations are specified independently from each other. In order to participate in an organization, agents are assumed to be able to reason about an organization. That is, there is

an expectation that the agents *understand* such specification. This expectation is in certain cases too strong, since agents implemented by third parties cannot necessarily be assumed to understand the specification of the organization. Instead, we assume that there are different ways this requirement can be met, for example, by making the agents organization-aware, i.e. program them to understand the organizational specification. Alternatively, a third party that is able to perform the organizational reasoning, such as a middleware agent or a service (e.g. an artifact [9]), can be made available. If agents have no knowledge about the details of the organization, a middleware would be responsible for delegating tasks, performing role enactment, coordinating, etc., while a service could be useful if the agents have some knowledge about the organization, but not enough to perform their own organizational reasoning.

Once agents are playing roles in an organization, there will be certain expectations of the agent; a role may have certain norms and objectives, and the organization expects that agents playing such role take part in achieving the objectives and adhering to the norms. This of course leads to the question: *should agents be able to deviate from the expectations?* If so, some kind of monitoring is needed to impose appropriate sanctions upon violating agents. The agents then need to decide who is responsible for monitoring and sanctioning in the system. If agents are not allowed to deviate, they should be somehow forced to achieve their objectives. However, since it is probably not possible to force agents to take certain actions, and certainly not ideal, *without* putting severe restrictions to their autonomy, we may instead restrict the kinds of agents in such system to agents that have a desire to fulfill their organizational objectives and that are capable of fulfilling them. In such case, we can still expect the organizational objectives to be achieved.

The dimensions discussed above provide us with the following kinds of systems:

Restricted-Middleware Agents are assumed to desire achieving the organizational objectives, i.e. the tasks they receive from the middleware. The middleware knows which role(s) an agent enacts, which objectives it is responsible for and whom it can (potentially) delegate tasks to. This knowledge is used to delegate the tasks to the agents in the system in a "step by step manner".

Restricted-Agent Agents in this system have beliefs about the organization, and uses those beliefs to decide which objectives to achieve, which to delegate, and how to coordinate with other agents. The agents are assumed to desire achieving the organizational objectives.

Regulated-Middleware The agents are not aware of the organization, so the middleware will delegate tasks in a "step-by-step" manner. Agents may now have own desires and can choose not to perform certain organizational tasks. Instead of letting the middleware proactively delegate tasks to the agents, the agents might decide by themselves if they want the middleware to delegate a task for them. Furthermore, the middleware should monitor and sanction agents violating their responsibilities.

Regulated-Agent Agents are able to reason about the organization, but are no longer (necessarily) assumed to desire the organizational objectives. This means that they at some points may violate their obligations toward the organization, and therefore risk being sanctioned. Without a middleware, monitoring and sanctioning is distributed among the agents in the system.

The middleware-approach is less complex than the agent approach, and the restricted approach is less complex than the regulated approach. Thus the least complex system is a restricted system with a middleware performing the organizational reasoning, while the most complex (w.r.t. these dimensions) is a regulated system in which the agents are aware of the organization and can reason about their objectives. In the present paper, we focus on objectives that the agents are assumed to desire fulfilling, so we will only consider the "restricted" systems.

4 Guiding Agents

The Blocks World for Teams (BW4T) environment (see e.g. [2]) is a simulated environment similar to the classical Blocks World planning problem. The environment consists of rooms connected to halls and a number of agents. Colored blocks are placed within each room and the agents are supposed to deliver a

subset of these blocks to a drop zone in a specific color sequence. Each room only fits one agent and the blocks in a room are only visible for an agent inside that room.

We use the BW4T scenario to simulate the first responders scenario by considering the drop zone being the ambulance, colored blocks being injured fans, and agents playing the roles of fans, medics and police officers. Fans are fighting just outside some of the rooms and they can stop the medic from rescuing injured fans by entering a room just before the medic does so. Police officers will look for areas where fans are standing, while medics will check the rooms to find injured fans. Agents in BW4T can communicate, which allows the agents to more easily coordinate and delegate their objectives. Medics can inform other medics about their progress toward locating or rescuing injured fans, police officers can inform each other about fights and coordinate the task of breaking them up, and medics can delegate the task of forcing blocking fans to move away to police officers that are more capable of this task.

4.1 Reasoning requirements

In the following we describe the aspects required for organizational reasoning. These aspects can be integrated in either an agent or a middleware. In the following when we talk about an agent reasoning about the organization, for simplicity we do not distinguish between agents reasoning themselves and a middleware reasoning on the agents' behalf, since we discuss the general concepts and not how they are implemented. Given an organizational specification, an agent should be able to decide which objectives it is responsible for (in terms of the roles it enacts) and whether these objectives are active (e.g., whether previous objectives in a landmark pattern are achieved). We call this the *option consideration*. The agent can then reason about committing to and achieving the objectives in the *organizational deliberation* phase.

If an agent chooses to attempt to achieve an objective by itself (with or without help from others), this means that the agent believes it has a plan which achieves the objective. This stems from the agent's capabilities, but how this is realized is out of scope of this paper. However, the matter of deciding whether or not to coordinate or delegate involves knowledge of the organization, thus requiring the agent to be able to reason about these concepts in relation to the specification of the organization in which it participates. We have identified the following situations in which agents should consider coordinating or delegating:

Delegation If role r_1 depends on role r_2 for an objective o, agents enacting r_1 should be able to reason about whether to delegate o to an agent enacting r_2. For instance, a medic should be able to reason about whether *blockingFanRemoved* should be delegated to a police officer.

Dependency coordination Conversely, agents enacting r_2 that have been delegated objective o should be able to reason that agents which are enacting r_1 should be informed when o is achieved.

Same objective If roles r_1 and r_2 are responsible for an objective o (note that r_1 and r_2 might be the same role), agents enacting either role should be able to reason that agents enacting the other will benefit from knowing if the objective has been achieved.

Same scene Agents participating in the same scene have the overall objective of achieving the objectives of the scene, i.e. ending the scene. This means that they should be able to reason that if other agents are participating in the same scene, they might benefit from knowing that certain objectives of the scene have been achieved.

Even though these considerations are relevant both when considering reasoning by agents and middleware, some of the situations might be difficult to handle when using a third party. For instance, if agents responsible for the same objective should coordinate, they need to know who to coordinate with, which is difficult if they have no organizational knowledge. Either the middleware should inform each agent which other agents to coordinate with, or the coordination should be done entirely by the middleware, requiring that it is able to perceive that an objective has been completed.

4.2 Building blocks for organizational reasoning

In the following, we illustrate how the building blocks of the organizational reasoning are implemented in the GOAL agent programming language [5]. GOAL is a goal-oriented agent programming language, which fits quite well with the notion of organizational objectives: by committing to an organizational objective, an agent simply adopts that objective as a goal. In GOAL programs are built by basic rules of the form **if** `msc` **then** `action` and **forall** `msc` **do** `action`, where `msc` is a mental state condition about the agent's beliefs and goals, and `action` are actions such as handles adoption, belief updates and sending messages. **if** -rules matches a single mental state condition, while **forall** matches all applicable mental state conditions. The mental atoms `bel` and `a-goal` succeed if the arguments are parts of the agent's beliefs or goals, respectively. The actions `adopt, insert, delete, send` are used for adopting goals, revising beliefs and communicating with other agents.

We show how a few of the most essential building blocks using the first responders scenario. In the following, `A, R, S, O` corresponds to an agent, role, scene and objective, respectively. If more than one of each is used, they are distinguished by a number. Anonymous variables are denoted with an underscore, `_`. We have simplified the code to be more easily comprehended. A part of the agent's main GOAL program is shown below, stating that if the agent believes it has an organizational option, it considers it. Otherwise, it updates its options: Following the discussion above, we define an option as an active objective that the agent (via its role) is responsible for and the agent updates its beliefs to include this option (the agent should also consider whether previous options are no longer believed as being options):

```
forall bel(option(A,O,S)) do organizationalDeliberation.
forall bel(rea(A,R,S), responsible(O,S,R), active(O)) then insert(option(A,O,S)).
```

The medics are responsible for locating and rescuing injured people and initially only *injuredLocated* is active, thus an agent enacting the role of a medic will update its beliefs to include the organizational option of locating injured people. A very simple medic might choose to adopt the goal of locating injured people if it is an option:

```
if bel(option(_,injuredLocated,_)) then adopt(injuredLocated).
```

It can continuously inform other agents responsible for locating injured people about its progress (i.e. the rooms checked and the injured people found):

```
forall a-goal(injuredLocated), bel(rea(A,R,S), responsible(injuredLocated,S,R)) do {
    forall <injured found> do send(A, <location>).
    forall <room checked> do send(A, <room>).  }
```

When the agent believes that all rooms have been checked, it believes that it has reached the landmark for the objective. Note that this activates the next landmark, *injuredRescued*, which will be available the next time the agent considers its organizational option.

```
if a-goal(injuredLocated), bel(checkedRooms) then insert(reached(injuredLocated)).
```

Whenever an objective of an option has been reached, the agent will stop believe that it is an option and inform other agents in the scene that the objective has been completed:

```
forall bel(option(A1,O,S), reached(O)) do {
    if true then delete(option(A1,O,S)).
    forall bel(rea(A2,_,S)) do send(A2, :reached(O, S)). }
```

The main difference between an agent and a middleware in this context is that the middleware performs the reasoning *on behalf of* the agents. This means that the middleware is considered an organizational agent, and the building blocks above can thus be used to implement a middleware as a GOAL agent. Note that letting someone perform reasoning on behalf of the agents have some implications. A middleware will have full control over how objectives are delegated to agents, so if an objective is represented by a set of sub-objectives that collectively complete the objective, the middleware can delegate the sub-objectives to individual agents, thus coordinating the fulfillment of the objective. For example, an objective of rescuing all injured persons can be divided into a sub-objective for each of the injured persons. The middleware

can then delegate a sub-objective to the medics in the system, e.g. by delegating each injured person to the nearest medic. On the other hand, the middleware has no or limited access to each agent's beliefs and desires, so taking the agent's state and preferences into consideration is difficult, if not impossible. Even if a medic currently has a personal desire to avoid rescuing a specific person because that person is located in the middle of the fight, the middleware is unable to consider this, and it might delegate the task to the medic anyway.

Agents reasoning about an organization are organization-aware, so each agent is responsible for its own reasoning w.r.t. the organization. This provides the agents with more freedom and more possibilities, since they can now consider their own beliefs, meaning that the reasoning can be more fine-grained, allowing them to consider an objective in different situations. Being able to consider desires has the advantage (from the agent's perspective) that the agent can ensure that it does not perform organizational actions that leads to undesirable situations (such as the one above, where the medic can choose to avoid certain locations). However, it has the disadvantage (from the organization's perspective) that certain objectives might never be fulfilled (e.g. if the fight never stops, and the medic refuses to rescue the injured persons at the undesirable locations). Thus it increases the agent's autonomy but also the risk that certain objectives might not be completed.

Since the building blocks are integrated in GOAL agents, it shows that it enables agents that are initially unable to reason about organizations, to use information about an organization to decide on how to achieve its organizational objectives, whether this is by coordination, delegation or something else. The assumption that agents wants to achieve the organizational objectives gives us assurance that they will eventually be achieved, and a direction of future research will be to investigate how to have the same kind of assurance when this assumption has been relaxed.

5 Conclusion

We have presented an approach for performing organizational reasoning by using building blocks that can be integrated in agents or used by a middleware. By establishing a number of dimensions of organizational reasoning, we have identified the requirements for agents reasoning about organizational objectives. We have argued that this reasoning can be naturally divided into two parts: option consideration and deliberation. In the consideration phase, the options (i.e. objectives) that are possible to commit to are found. In the deliberation phase, the agent uses its knowledge about the organization to determine how to complete its objectives, either by itself or by coordination and delegation. We have shown that the basic building blocks can be applied to both organization-aware agents and third parties such as a middleware performing the reasoning on the agents' behalf.

The approach in this paper is based on the OperA model, but apart from OperA, there are other organizational models, such as ISLANDER (part of the EIDE framework [3]) and $\mathcal{M}$OISE$^+$ [7]. It is therefore quite natural to consider how to let agents reason about these models in order to make them practical. While this paper focuses on building blocks that can be used both by agents performing the reasoning themselves and by a middleware, other approaches usually considers how to incorporate organizational reasoning by use of a middleware or artifacts. AMELI (also part of the EIDE framework) and $\mathcal{S}$-$\mathcal{M}$OISE$^+$ are agent-based middleware solutions for ISLANDER and $\mathcal{M}$OISE$^+$, respectively. The organizational agents of these systems ensure that the organization remains consistent and that no prohibited organizational actions are performed. Artifacts are devices that provide certain functionalities agents use by interacting with them. In [9] it is suggested that *boundary artifacts* can be used to control the presence of agents inside an organization and to enact the contract between the agents and the organization. This is further elaborated in [6] where ORA4MAS (Organizational Artifacts for Multi-Agent Systems) is presented. ORA4MAS is a general approach suitable for different kinds of organizational models. They argue that middleware solutions typically takes away the control from the agents, whereas artifacts brings this control back to the agents, since the agents can, via their autonomy, choose whether or not to interact with the organizational artifacts of the system. In [6] it is integrated with $\mathcal{M}$OISE$^+$ and it is shown how that model can be implemented using artifacts.

Our solution enables both approaches, depending on how much information the agent discloses to the middleware. If nothing is disclosed, the middleware will have to make very general decisions and can mostly provide "step-by-step" guidance, while if the agent discloses everything, a middleware will be able to perform organizational reasoning resembling that of an organization-aware agent. We argue that the ultimate way of bringing the control back to the agents is to allow the agents *themselves* to perform the organizational reasoning. Our building blocks allow agents to perform this kind of reasoning, either by disclosing their beliefs and desires to a middleware, or by integrating the building blocks in the agents.

Due to space limitations, we have only presented a few building blocks. It is clear that different strategies exist in both the consideration and deliberation phase, allowing organization-aware agents to perform organizational reasoning that is more sophisticated. For example, an agent might not only consider active objectives as options, if it wants to look ahead, and it might want to consider objectives, that it itself is not directly responsible for. We have left an investigation of these strategies as a direction for future research.

Implementing the reasoning inside an agent or a middleware corresponds to situations in which either the agents are organization-aware or a middleware handles all the organizational reasoning, but hybrid situations may also occur. For example, if some agents are organization-aware, while others are not, or if some of the agents are only partially organization-aware. A number of issues arise in such systems: how can organization-aware agents coordinate with "unaware" agents and how can the middleware know that certain agents are aware of the organization, and therefore not perform the reasoning for them? Furthermore, a partially organization-aware agent needs to look to the middleware for support when it is stuck. In the future we will investigate how to handle such situations, since in open societies these different types of agents may very well co-exist.

References

[1] H. Aldewereld and V. Dignum. OperettA: Organization-oriented development environment. In *Languages, Methodologies, and Development Tools for Multi-Agent Systems*. Springer, 2011.

[2] H. Aldewereld, V. Dignum, C. M. Jonker, and M. B. van Riemsdijk. Agreeing on role adoption in open organisations. *KI - Künstliche Intelligenz*, 26(1), 2012.

[3] J. L. Arcos, M. Esteva, P. Noriega, J. A. Rodríguez-Aguilar, and C. Sierra. An integrated development environment for electronic institutions. In *Software Agent-Based Applications, Platforms and Development Kits*. Birkhäuser, 2005.

[4] V. Dignum and F. Dignum. Modelling agent societies: Co-ordination frameworks and institutions. In *Progress in Artificial Intelligence, LNAI 2258*. Springer-Verlag, 2001.

[5] K. V. Hindriks. Programming Rational Agents in GOAL. *Multi-Agent Programming: Languages, Tools and Applications*, 2, 2009.

[6] J. F. Hübner, O. Boissier, R. Kitio, and A. Ricci. Instrumenting multi-agent organisations with organisational artifacts and agents. *Autonomous Agents and Multi-Agent Systems*, 20(3), 2010.

[7] J. F. Hübner, J. S. Sichman, and O. Boissier. S-moise+: A middleware for developing organised multi-agent systems. In *COIN I, volume 3913 of LNAI*. Springer, 2006.

[8] S. Kumar, M. J. Huber, P. R. Cohen, and D. R. Mcgee. Toward a formalism for conversation protocols using joint intention theory. *Comp. Intelligence*, 18, 2002.

[9] A. Ricci, M. Viroli, and A. Omicini. Programming MAS with artifacts. In *Proceedings of the Third international conference on Programming Multi-Agent Systems*, ProMAS'05, Springer, 2006.