Towards Organization Aware Agent-based Simulation

Maarten SIERHUIS, Catholijn JONKER, Birna van RIEMSDIJK and Koen HINDRIKS

Abstract—No organization exists without actors. organizations whether formal or informal are the way people coordinate their activities and collaborate. The dynamics of organizations are caused by the dynamics of its actors and the environment of the organization. In this paper we layout a generic framework for agent-based simulations that allows the modeller to simulate the extent to which agents are aware of the organizations they are part of and the extent to which the agents conform their behavior to the policies and norms of the organization. Furthermore, the framework allows agents to join or leave organizations and to take up or drop roles within the organization. We formulate the requirements for organization aware agent-based simulation, we provide a basic architecture in the agent-based simulation environment Brahms, and we illustrate the principles in a case study based on a partial MOISE$^+$ specification of a soccer team.

Index Terms—agent-based modeling and simulation (ABMS), multi-agent systems (MAS), organization, organizational modeling (OM), roles, groups, joint activity

1. INTRODUCTION

ORGANIZATION and activity aware agents are important for developing *open heterogeneous* multi-agent systems (MAS). Developing and experimenting with open MAS is still in its infancy. For that reason we are working towards simulations of such systems in which organizations are specified explicitly, and the agents involved are organization aware. Awareness is necessary because in open heterogeneous MAS, software agents, people, systems and robots can come together to form teams, and work as part of an organization that either exists formally or is ad hoc. The issues are plentiful. The MAS community researches some of these issues, such as open organization modeling [1], [2], [3], argumentation frameworks [4], [5], [6], [7], teamwork [8], [9], [10], [11], and culture [12], [13], [14]. However, there does not exist one framework that unifies all these theories into a framework for modeling organization and activity aware agents. In this paper, we discuss what is needed for developing a theoretical framework for modeling and simulating organization and activity aware agents. In such a framework, agent modelers need to be able to represent the different aspects of organization, individual and teamwork behavior, norms, policies and culture of the system and ways common ground is achieved in the system, in such a way that the agents can reason with these concepts and

can dynamically change the current structures represented with these concepts in their individual belief system, as well as in the system's state and environment. We argue that a multiagent simulation environment is needed to simulate agents coming into and leaving the environment, in such a way that the issue with regards to open heterogeneous MAS can be researched. Indeed, we are proposing agent directed simulation [15] as a tool for researching open heterogeneous MAS.

Modeling an open heterogeneous environment has the following aspects:

- **Organization aspect**: How do you manage the dynamic entry and exit of actors within the organization(s) and the environment? For this, actors that enter an organization need to be able to dynamically get an understanding (i.e. learn) of the groups, roles and responsibilities in the organization. It is important to note that actors could be software agents, external systems (even robots) and people.

- **Behavioral aspect**: Actors that become part of an organization need to be able to understand (i.e., learn) and reason not only about the role(s) it and others are playing, but also what it means to play that role. In other words, the actor needs to be able to learn and reason about how to fulfill the role(s) it is playing and about the dependencies between this role and others in the organization. For the behavioral aspect we combine modeling behavior in terms of goals and tasks, with modeling activities and work practice [16], [17].

- **Collaboration aspect**: Besides any formal and informal organizational structures, there exist many different teams or adhocracies[1] through which actors perform their individual and joint activities. The actors need to be able to form teams and do work within the team, and thus they need to be able reason about the teamwork aspect.

- **Regulation aspect**: To ensure smooth operations, every organization has norms, policies, laws (in terms of permissions and obligations), and an organizational culture. The better the agents in the organization understand and adhere to these regulations, the smoother operations will be. This implies that agents entering the organizations need to learn and understand the regulations, to decide to what extent they will adhere to them, as well as consider the monitoring and detection of individual actor and team violations of norms and policies and their punishment.

- **Common Ground aspect**: Common ground is the ability

---

[1]An adhocracy is a flexible, adaptable, and informal organizational structure without bureaucratic policies or procedures.

of actors to debate and come to a shared understanding of positions and arguments. In order for actors to work together they need to be able to create common ground [18] .

- **Environment aspect**: All organizations are implemented in an environment. Human and robot organizations are implemented in physical places and spaces/structures, dealing with physical events, while software agent organizations are on the one hand implemented in physical objects (i.e., computers), but behave within a virtual environment (i.e., computer memory and communication networks). In all cases, the agents within the organization need an ability to use, reason and communicate about the environment [19].

In this paper, we present how the above aspects can be combined in a modeling and simulation architectural framework. We focus on the organization and environment aspects, and to some degree on the behavioral and collaboration aspects. The regulation, common ground, and collaboration aspects will be addressed in future work in which the agents will deliberate on their options regarding these matters. We present a first implementation of the framework in the Brahms agent simulation environment. Brahms is an open agent framework where both agents developed in the Brahms agent-oriented language (AOL) and agents developed in the Java language can be executed [20], [21]. The Brahms environment can run in either simulation mode or in realtime mode. In realtime mode, a Brahms application is a MAS. In simulation mode, a Brahms application is a discrete-event multi-agent simulation of agent and object execution (representing people, software agents, robots, artifacts or external systems), agent and object interaction, and agent organization and environment.

The rest of the paper is divided into the following sections. Section 2 provides an example about a soccer team that we use to apply the framework from section 4. Section 3 gives background on ideas and concepts related to organizational modeling. Section 4 describes the organizational modeling and simulation framework, based on MOISE$^+$. We end the paper with a conclusion in section 5. The appendix provides some Brahms soccer model source code, and provides URLs to download run the model yourself.

2. Leading example: soccer

To illustrate our ideas we work with examples that are based on soccer. First of all we show a generic organization model for soccer that is based on the MOISE+ soccer team organization structure in Hübner, et al. [22]. It defines what a soccer team is, its roles, an attack scheme that is part of the team's global organizational goal, and the missions to support the attack scheme formulated at the level of the roles. The first of those role missions is to get the ball, go towards the opponent field, and pass the ball an agent that is placed in the midfield and that is committed to the second mission, i.e. to dribble the ball to the opponent back line and then to pass the ball to an agent that is in the opponent goal area and is committed to mission of shooting the ball at the opponent's goal. Although the generic structure is quite clear,

from the point of view of the agents fthat are supposed to realise the global organizationals goals, it is not so easy for them to perform their missions. Everyone slightly familiar with team sports knows that it is not enough to assign agents in a static way to such missions. On the one hand, agents are assigned to roles, however, during the game, the agents can dynamically and temporarily take over roles when the situation demands it. Furthermore, the agents will have to observe without communication, which agents are committed to the missions in the plan (as communication is often of limited value in a real game environment). Finally, it is an agent's judgement if passing the ball to an agent, committed to the mission next in the plan, is actually in the position to carry out that mission; is there no opponent agent that is likely to intercept the pass? This requires an agent that understands not only the roles played by its teammates, but also, the behavior and capabilities of the opponent team members, which is part of ones work practice. This example, at the very least, illustrates the following requirements of the organization; be it the agents or the organization specification:

- agents have to:
  - understand their own roles
  - understand, to the point of correct anticipation/prediction, the roles of agents that they might have to pass the ball to
  - understand the roles of agents that they might have to temporarily take over
  - be able to trust team mates to take over its own role when necessary
  - understand the behavior (or roles) of opponent agents so that they can anticipate on that behavior
  - have the capabilities needed to observe and correctly interpret the dynamic situation, and the capabilities to exercise their own role and, to some extent, other roles
- the organization should enable and/or encourage all of the above.

The second example from the soccer domain considers a specific instantiation of a well-known soccer team in The Netherlands. The case study presented here is hypothetical, but the reference to well-known soccer players and trainers is deliberate. The leading characters are

- Sjakie Meulemans, a talented young central forward that has just transferred from another soccer team FC Volendam
- Rinus Michels, the coach of Ajax
- Johan Cruijff, the current center forward of Ajax

Amsterdamsche Football Club Ajax, referred to simply as Ajax, is a professional football club from Amsterdam, The Netherlands. Ajax raises good football[2] talent that's often sold for large sums of money to other European clubs. These talents are brought on at an early age. Sjakie Meulemans is such a talent. Sjakie is a young player in the youth team of Ajax. He has just been hired away from the youth of FC Volendam, not

---

[2]Although we adhere to the more accepted term football, in this paper we continue with the use of the term soccer to stay consistent with the MOISE$^+$ example we use.

only because of his speed, but mostly because of his incredible ability with the ball (his foot work). Sjakie is only seventeen and is a young and talented center forward, whose talent lays in scoring goals in the most unbelievable ways. Some talk about Sjakie's "wondersloffen" (transl: miracle shoes). The young-talent scouts from Ajax spotted him on a wet Saturday, six months ago, on the fields of FC Volendam. They spoke to the coach of Ajax, showed him some video of Sjakie, and with his approval started negotiation with FC Volendam. Sjakie is thrilled to now start playing for this illustrious team.

As Sjakie has been playing soccer since the age of 5, he is well familiar with the generic organizational structure of soccer teams, and that of FC Volendam in particular. Furthermore, Sjakie is well familiar with Ajax and its famous soccer players. However, Sjakie needs much more information on how things are organised at Ajax. Also he is still unsure which position he will play in the team.

Today, Sjakie is joining the team for the first time in a practice session. Coach Rinus Michels has high hopes for Sjakie to play central forward for Ajax as well. Before Sjakie arrives, Rinus makes it clear to the team that he wants to test Sjakie's talents in the next game, and that he wants them to treat him as one of the team. He informs them that in that next game Sjakie will play the central forward role. On the question of Johan Cruijff, the current central forward, Rinus explains that Johan will have to play as a left forward in the next game, a position Johan is very familiar with. Furthermore, Rinus asks Johan to explain to Sjakie what is expected of him during the game, i.e., the different activities and tasks he needs to perform as a central forward in Ajax and what he can expect of the players around him.

As Sjakie walks into the dressing room, he meets all the players for the first time. One by one he shakes their hand, as they explain what position they are playing in the team. After they are all dressed for the training, they walk together to the training field. Coach Rinus Michels tells everyone to go for a warmup run, while he talks to Sjakie. Rinus explains to him the organization of the team and Sjakie's role within it. Michels tells Sjakie to talk to Johan during the training, because he will explain to Sjakie the different activities and plays he should learn for the next game. Sjakie knows how to be a forward, i.e., he knows he needs to score goals, but he realizes that his role within the Ajax squad could be of a different nature. Does he need to help defend? Does he have a man to cover? Does he need to fall back to the mid-field to accept the ball on an attack, and to which of the wing players, left or right, does he need to pass the ball when he receives it? Etcetera, etcetera. During the training session, Johan stays with Sjakie and at appropriate moments he talks to Sjakie and gives him the specifics he needs to know for his role as a forward in the attack scheme they are going to play in the next game. These discussion continue after the training and over the next days. During the training sessions Sjakie and the other team members learn to interpret each others behavior and capabilities. The old team members find out that Sjakie is a dedicated and versatile player. He is always in time, polite to everyone, and attentive to instruction and tips. During the next game, Johan who knows Sjakie best, succesfully passes

the ball to him, and Sjakie scores a goal.

The above scenario illustrates some aspects of team play and organizations for which various solutions could be found and only one is contained in the scenario:

- new team members have to be informed of / have to learn
  - the organization of the team: structure, role specifications, and regulations
  - the role assignment to agents
  - the capabilities of the other team members
- existing team members have to be learn
  - the role that the new team member is going to play
  - the capabilities of the new team member
  - the attitude of the new team member towards the regulations of the organization

Note that in settings such as these, the integration of new team members will not always go so smoothly. Not everyone has such a generous heart and the self-confidence of Johan, who had to give way to Sjakie at least for the time being. Such aspects are not discussed further in this paper. However, they are important, because the emotions of team members may make them behave in unpredictable, or even obstructive ways, violating various regulations of the organization [23].

The above examples will be addressed in the following sections to illustrate our framework and an example instantiation in Brahms.

3. BACKGROUND

In this section, we present the main ideas and techniques upon which we base our organizational simulation framework in Brahms. Our aim is to be able to model the full *richness* of organizations and the agents that populate them. In particular, we aim to facilitate both the top-down *explicit* specification of organizational aspects such as organizational structure, as well as bottom-up *coordination driven* by the agents of an organization. These two perspectives are called organization-centered and agent-centered, respectively, in [24]. A second dimension along which agent organizations can differ, is whether agents are aware of the organization in which they operate [24]. Being aware allows the agents to reason with and about the organization.

All of these aspects of agent organizations are already challenging in and of themselves. Therefore, it is not the aim of this paper to address any of these aspects in depth. Rather, we take a first step towards investigating how they can be integrated to support rich organizational modeling and simulation using Brahms. In the rest of this section, we first provide general background on agent-based organizational simulation, and describe existing work on which we build for organizational simulation in Brahms.

3.1. Organizational Modeling and Simulation

Within Computational organization Theory and Artificial Intelligence, a number of organization modeling approaches have been developed to simulate and analyse dynamics within organizations; e.g. [25], [1], [26], [27], [24]. Some of these approaches explicitly focus on modeling organizational structure, abstracting from the detailed dynamics. Other approaches

put less emphasis on organizational structure but focus on the dynamics in the sense of implementing and experimenting with simulation models. These simulation models are based on some implementation environment that has no dedicated concepts for organization simulation/implementation. The Strictly Declarative modeling Language SDML [26], and the use of the agent-oriented modeling approach DESIRE in social simulation as presented in [28] are exceptions. Both modeling approaches focus on specification and simulation; however, they do not offer dedicated support for modeling and simulating organizations, let alone implementing open organizations for agents on the Internet.

Organizational modeling languages support the explicit specification of organizations using the notion of "role" [29], [27], [30], [24], [22], [31], [32]. In this way, an organizational specification *abstracts* from the individual agents that will eventually play the roles. Organizational modeling languages allow to model various aspects of an organization, such as its structure, work processes and norms.

The idea is that an explicit organizational specification can be used to *organize and regulate* collections of autonomous agents in order to make them more effective in attaining their purpose or to prevent certain undesired behavior from occurring. For example, norms, laws and policies can be seen as constraints imposed by society on the behavior of the individuals or agents [33].

Although agents are expected to adhere to these constraints, they might nevertheless decide to violate them. The violation of a norm, if noticed by other agents, can cause the violating agent, for example, to suffer a loss of reputation and/or rebukes. With respect to monitoring and punishing, the organization might deploy machinery and/or agents. For example, camera's and police patrols are used to monitor speeding of cars. Using agents for monitoring norm violation has been explored in the context of Brahms in [34]. It can also be the case that violation is prevented by constraining the set of executable actions of the agents, e.g. by setting up physical restraints of getting on the metro if the agent does not have a ticket.

Some organizational modeling languages come with implementation frameworks [35], [22], [36] 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. Organizational modeling languages allow to model various aspects of an organization, such as its structure, work processes and norms that should be adhered to by the agents of the organization.

In real organizations, roles can change and/or new roles can be created without those changes being reflected immediately in the organization specification. In such cases, when the role changes and new roles are deemed effective by the organization, the organization specification is updated accordingly. Thus, when modeling and simulating organizations we insist on an implementation framework that allows for such self-organizational behavior.

In this paper, we base the modeling of an organization in Brahms on the well-known MOISE$^+$ organizational modeling language [24], [22]. MOISE$^+$ 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. MOISE$^+$ comes with an organizational middleware that allows agents to access and modify the state of the organization, and ensures that certain organizational constraints are respected. For example, if the structural dimension specifies that an agent cannot play two particular roles at the same time, the middleware will prevent an agent from committing to playing both of these roles.

The MOISE+ middleware [24], [22] is an important first attempt to separate organization modeling and specification from the design and implementation of the agents that populate organizations. This work differs from ours in that the framework of MOISE+ is translated in an adhoc way to the agents, thus every agent has to be modeled and implemented for the application at hand. Both in MOISE+ and KAoS [37], [38] the attempt so far have been to directly, but external to the agent, determine the behavior of the agents that play a role in the organization. In particular, agents have no choice but to behave according to the role specification of the role they are playing. In the framework presented in this paper, the agents can determine amongst themselves, or together, whether or not to play certain roles in the organization and commit to missions and norms.

3.2. Organizational Reasoning

If agents are aware of the organization of which they are a part, this allows them to reason with and about the organization. For example, an agent may be aware of a norm like "one should always give an answer to a request", but he might not agree with it. This means he does not necessarily comply with the norm, and if he does it might be an unconscious decision or because it benefits him. He can exploit the existence of the norm, for example, by always first requesting information before trying to find it himself. The agent can also accept a norm. In that case he agrees that the norm is a good one and he tries to follow it as much as possible. Only in special situations (e.g. if the norm contradicts another, more important norm) will he violate it.

Reasoning explicitly about norms is included in a number of social simulations, see, e.g. [39], [40]. In this paper, we don't focus on sophisticated organizational reasoning, but we show how agents can understand an organizational specification on which one can build sophisticated reasoning.

4. ORGANIZATIONAL MODELING FRAMEWORK

In this section we describe our organizational modeling framework for the Brahms language, based on MOISE$^+$. Even though we use Brahms as the agent-based modeling language, the framework is general and can be implemented in other agent-oriented modeling languages that support the capabilities needed for modeling the framework concepts (i.e., multiple agents, groups, classes, objects, geography, activities, goals, and communication). We use a some-what modified UML form to provide the Brahms models. In some cases,

where we feel it helps the reader, we provide some Brahms source code.

4.1. Organization aspect

A MOISE$^+$ structural specification consists of a set of roles and a set of groups, and the relations between them. Roles can be related through an inheritance relation, and a role or group can be related to another group through an "is part of" relation. Roles can also be related through several kinds of links: an acquaintance link, a communication link (agents are allowed to communicate if they play roles with a communication link), an authority link, and a compatibility link (an agent is not allowed to play two roles at the same time that are not compatible). For every authority link there is an implicit communication link, and for every communication link there is an implicit acquaintance link.

In Brahms, we model a MOISE$^+$ structural specification in two phases. To model the MOISE$^+$ organizational aspect in Brahms we have to separate two parts of the organization structural dimension. First, the agents need to be able to use concepts defining a MOISE$^+$ organization structure (see fig. 1). Second, the agents need to understand and be able to reason with the instantiation of these concepts for a specific type of organization, e.g. a soccer team (see fig. 2).

4.1.1) Modeling the MOISE$^+$ structural dimension: Simply speaking, the Brahms MOISE$^+$ organization structure defines an organization in terms of the *roles*, *groups* and *links* concepts, whereas an instantiation of this representation defines a specific organization of its kind, with Brahms classes and conceptual objects for each MOISE$^+$ role and group known in the organization. If we want agents to be able to dynamically enter such an organization, the agent needs to learn about both the concepts (i.e., the roles, groups and links) defined in the organization structure and the instantiation of these concepts for the specific organization it enters.

Figure 1 shows the UML class diagram for the MOISE$^+$ concepts in the structural dimension as they are modeled in the Brahms language. Every agent within the organization needs to know these concepts (i.e., classes or types) in order to reason about specific instantiations of these concepts for a particular organization. There are three general MOISE$^+$ concepts shown in figure 1 that are defined as conceptual classes in Brahms; *MoiseOrganizationStructureConcept* (this is a derived class), *MoiseRoleConcept* and *MoiseGroupConcept*. Every MOISE$^+$ concept in a specific instance of an organization structure inherits from one of these two classes. The *MoiseRoleConcept* class defines the relations for roles in a MOISE$^+$ organization structure, while the *MoiseGroupConcept* class defines the relations for groups in a MOISE$^+$ organization structure. The MOISE$^+$ role link concepts are modeled as relations within the *MoiseRoleConcept* class. MOISE$^+$ discriminates the role links *authority over*, *communication with*, *compatible with*, and *acquaintance of*. In figure 1 these are shown as *MoiseRoleConcept* relation attributes. Similarly, the MOISE$^+$ group relations *has subgroup* and *has member roles* are shown in figure 1 as relation attributes.

Figure 2 shows the UML class diagram for an instantiation of figure 1, namely a MOISE$^+$ soccer team organization
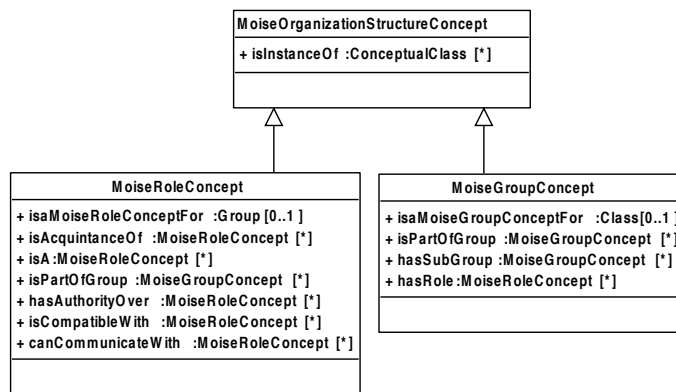
Fig. 1.   Concepts for modeling MOISE+ organization structure

structure. First, we define classes for both *SoccerTeamRole* and *SoccerTeamGroup*, after which we define the object instances representing the role and group concepts for a soccer team.
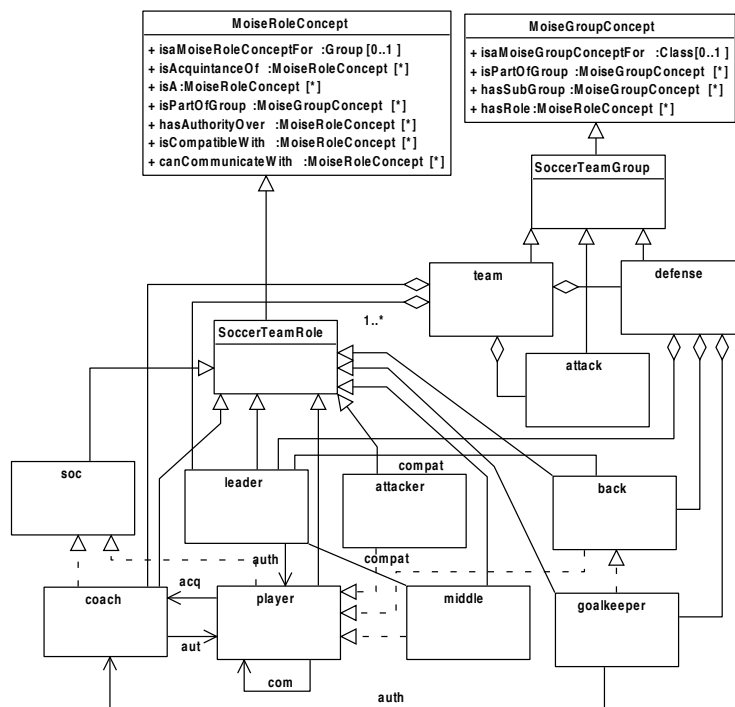
Fig. 2.   Concepts for MOISE$^+$ soccer team organization structure

Having these soccer team roles and groups defined does not mean that the agents coming into this organization automatically know about these concepts, or can reason with them. For this, something needs to happen. Our framework is flexible enough to define different ways for incoming agents to learn about the organization structure. Maybe we want to model that a new "hire" gets a "document" to read to understand the roles, groups and links in the organization, or someone tells him or her the information in a formal "intake" session, or he or she gets pointed to a website. We might want to decide that every newly created agent automatically knows about the soccer team organization structure. Of course, we also want to give agents the capability to define a new MOISE$^+$ organization

structure out of nowhere, or change an existing one.

```
object MoiseSoccerTeamStructureDoc instanceof
        MoiseOrganizationStructure {
  initial_beliefs:
    (current.describesMoiseOrganizationStructureOf =
        SoccerTeam);
    (soc isInstanceOf MoiseRoleConcept);
    (soc isInstanceOf SoccerTeamRole);
    (soc.isaMoiseRoleConceptFor = SocRole);
    (player isInstanceOf MoiseRoleConcept);
    (player isInstanceOf SoccerTeamRole);
    (player isA soc);
    (player.isaMoiseRoleConceptFor = PlayerRole);
    (coach isInstanceOf MoiseRoleConcept);
    (coach isInstanceOf SoccerTeamRole);
    (coach isA soc);
    (coach.isaMoiseRoleConceptFor = CoachRole);
    (back isInstanceOf MoiseRoleConcept);
    (back isInstanceOf SoccerTeamRole);
    (back isA player);
    (back.isaMoiseRoleConceptFor = BackRole);
    (middle isInstanceOf MoiseRoleConcept);
    (middle isInstanceOf SoccerTeamRole);
    (middle isA player);
    (middle.isaMoiseRoleConceptFor = MiddleRole);
    (attacker isInstanceOf MoiseRoleConcept);
    (attacker isInstanceOf SoccerTeamRole);
    (attacker isA player);
    (attacker.isaMoiseRoleConceptFor = AttackerRole);
    (goalkeeper isInstanceOf MoiseRoleConcept);
    (goalkeeper isInstanceOf SoccerTeamRole);
    (goalkeeper isA back);
    (goalkeeper.isaMoiseRoleConceptFor = GoalKeeperRole);
    (leader isInstanceOf MoiseRoleConcept);
    (leader isInstanceOf SoccerTeamRole);
    (leader isA player);
    (leader.isaMoiseRoleConceptFor = LeaderRole);
    (attack isInstanceOf MoiseGroupConcept);
    (attack isInstanceOf SoccerTeamGroup);
    (attack.isaMoiseGroupConceptFor = AttackGroup);
    (defense isInstanceOf MoiseGroupConcept);
    (defense isInstanceOf SoccerTeamGroup);
    (defense.isaMoiseGroupConceptFor = DefenseGroup);
    (team isInstanceOf MoiseGroupConcept);
    (team isInstanceOf SoccerTeamGroup);
    (team.isaMoiseGroupConceptFor = TeamGroup);
    (attack isPartOfGroup team);
    (defense isPartOfGroup team);
    (team hasSubGroup attack);
    (team hasSubGroup defense);
    (team hasRole leader);
    (team hasRole coach);
    (attack hasRole leader);
    (attack hasRole middle);
    (attack hasRole attacker);
    (defense hasRole goalkeeper);
    (defense hasRole back);
    (defense hasRole leader);
    (goalkeeper hasAuthorityOver back);
    (coach hasAuthorityOver player);
    (leader hasAuthorityOver player);
    (player canCommunicateWith player);
    (leader isCompatibleWith back);
    …
```

Fig. 3.   Brahms code of MOISE$^+$ soccer team organization structure

To do this, we separate the definition of the MOISE$^+$ organization structure concepts and the soccer team roles and groups from the knowledge that agents can get about these defined concepts. Figure 3 shows the Brahms code of the definition of the knowledge about a MOISE$^+$ soccer team organization structure, as an object instance—named *MoiseSoccerTeamStructureDoc*—of the class *MoiseOrganizationStructure*. This object contains the knowledge (in terms of

beliefs stored inside the object) about the soccer team structure. Any team member agent can now "read" the information stored in this object, at which point the agent gets the beliefs stored in the objects. How and when the agent read this object can be defined in many different ways in the Brahms model. The point is that the agent gets the knowledge about the organization structure only when it has reads the information from the object.

Together, figure 1, figure 2, and figure 3 implement figure 3 (Soccer team structure using MOISE$^+$) from Hübner, et al.[22].

4.1.2) Instantiating the Ajax Soccer Team Organization: Now that we have defined all the MOISE$^+$ concepts in the Brahms model, we can instantiate the model for the Ajax Soccer Team example. First we need to create the Ajax Soccer Team organization model. This is done by creating object instantiations for the Ajax organizational concepts, as defined by the MOISE$^+$ soccer team organization structure in figures 2 and 3. There are two parts to a MOISE$^+$ organization that need to be instantiated; groups and roles. Figure 4 shows the three MOISE$^+$ group objects for Ajax. There as an *AjaxTeam* object that represents everyone that is part of the Ajax soccer team. Then there is an *AjaxDefense* and an *AjaxAttack* group object. These objects represent the defense and attack groups of the Ajax soccer team.
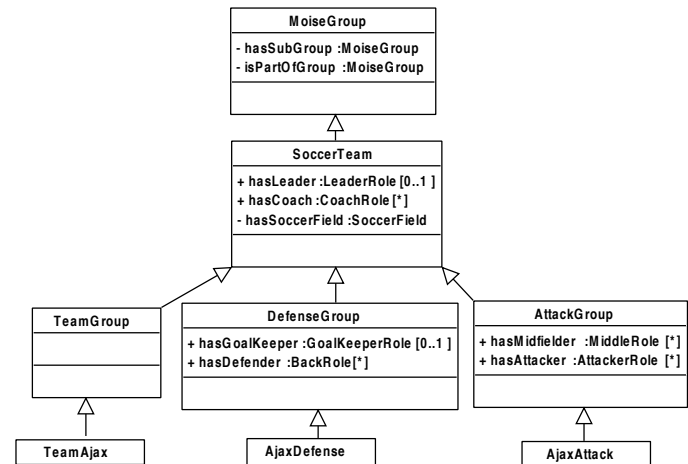
Fig. 4.   MOISE$^+$ groups and objects for the Ajax organization structure

We also need to create all the agents in the Ajax organization, i.e., the coach and all player agents. Figure 5 shows the coach and four player agents involved in our example. Agent *Rinus_Michels* is the coach (CoachRole). The current Ajax players include agents *cm_1* (central midfielder—MiddeRole) and *lm_1* (left midfielder—MiddleRole), and agent *Johan_Cruijff* who can play both as an attacker or as a midfielder (AttackerRole and MiddleRole). Agent *Sjakie_Meulemans* is the new forward (AttackerRole) of Ajax, who is joining the team. When Sjakie joins the team, coach Michels wants Sjakie to play on Johan's original forward position (left forward) and wants Johan to play left midfield, which means that left midfielder lm_1 has to sit out the next game. This is how the scenario starts.
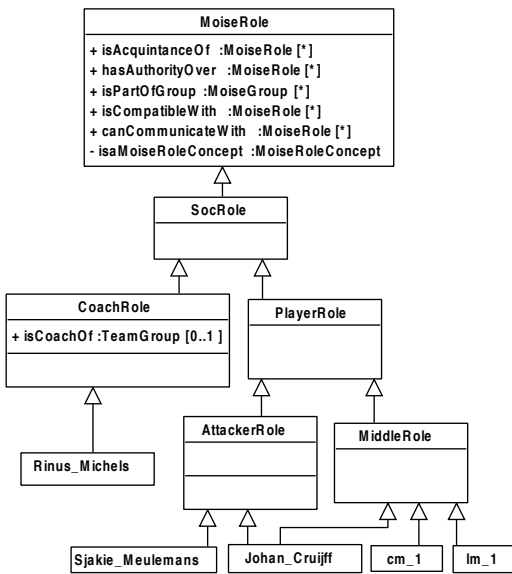
Fig. 5. MOISE$^+$ roles and agents for the Ajax organization structure

In order to play their roles, the agents need to have knowledge about the Ajax team organization, based on the organization structure as defined in figures 3-5. This is represented in figure 6. How do the agents get this knowledge? As mentioned before, this can be done in different ways. In our example, they get this knowledge by "reading" the *MoiseSoccerTeamStructureDoc* from figure 3. However, this is general knowledge that all soccer agents know. What about the Ajax specific team organization knowledge, partially represented in figures 4-6? Let us assume that there is another document, called the *AjaxFormalOrganizationDoc*, and the coach and the current Ajax player agents also have knowledge of this document, by having read it (see figure 7). Figure 7 shows the implementation in Brahms of the *MoiseSoccerTeamStructureDoc* object, shown as a UML diagram in figure 6. However, the new player, agent *Sjakie_Meulemans* has not yet read this document.

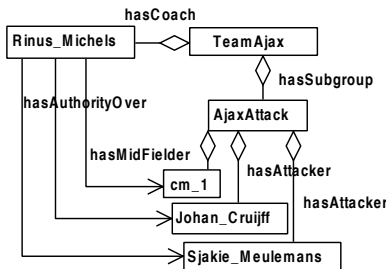Fig. 6. MOISE$^+$ agents and their relations for team Ajax

In our Ajax soccer example, agent *Rinus_Michels*, the coach, will have all the information about the Ajax organization as beliefs when he talks to agent Sjakie, thus communicating the information about the soccer team organization. Agent *Sjakie_Meulemans* thus learns about the organization from the coach agent. As mentioned before, we can model different ways for an agent to find out about the organization structure.

All *MoiseRole* agents will be able to reason over its beliefs

```
object AjaxFormalOrganizationDoc instanceof
       MoiseOrganizationStructure
{
    initial_beliefs:
      (current.describesFormalOrganizationOf = TeamAjax);
      (TeamAjax.hasSoccerField = Arena);
      (AjaxAttack isPartOfGroup TeamAjax);
      (AjaxDefense isPartOfGroup TeamAjax);
      (Rinus_Michels isaMoiseRoleConcept coach)
      (TeamAjax hasCoach Rinus_Michels);
      (cm_1 isaMoiseRoleConcept player);
      (cm_1 isaMoiseRoleConcept middle);
      (cm_1 isPartOfGroup AjaxAttack);
      (lm_1 isaMoiseRoleConcept player);
      (lm_1 isaMoiseRoleConcept middle);
      (lm_1 isPartOfGroup AjaxAttack);
      (Johan_Cruijff isaMoiseRoleConcept player);
      (Johan_Cruijff isaMoiseRoleConcept leader);
      (Johan_Cruijff isaMoiseRoleConcept middle);
      (Johan_Cruijff isaMoiseRoleConcept attacker);
      (Johan_Cruijff isPartOfGroup AjaxAttack);
      (Sjakie_Meulemans isaMoiseRoleConcept player);
      (Sjakie_Meulemans isaMoiseRoleConcept attacker);
      (Sjakie_Meulemans isPartOfGroup AjaxAttack);}
} //end AjaxFormalOrganizationDoc
```

Fig. 7. AjaxFormalOrganizationDoc object in Brahms source code

```
group MoiseRole memberof MoiseBaseGroup {
…
thoughtframes:
    thoughtframe tf_WhoHasAuthorizationOverWhom {
      variables:
        foreach(MoiseRoleConcept) roleconcept1;
        foreach(MoiseRole) roleinstance1;
        foreach(MoiseRoleConcept) roleconcept2;
        foreach(MoiseRole) roleinstance2;
      when(knownval(roleconcept1 hasAuthorityOver roleconcept2)
       and knownval(roleinstance1 isaMoiseRoleConcept roleconcept1)
       snd knownval(roleinstance2 isaMoiseRoleConcept roleconcept2)
       and knownval(roleinstance1 != roleinstance2))
      do {
        conclude((roleinstance1 hasAuthorityOver roleinstance2));
      }//do
    }//tf_WhoHasAuthorizationOverWhom
}//MoiseRole
```

Fig. 8. MOISE$^+$ Role Authorization Rule in Brahms source code

about the MOISE$^+$ concepts. For example, in order to decide who in the organization has authority over whom, the agents inherit the production rule from figure 8. This rule spells out that if a MOISE$^+$ role concept, defined for a specific domain, has authority over another MOISE$^+$ role concept (see figure 2) and an agent1, instance of the *MoiseRole* group, is playing the role of that role concept, and a second agent2 is playing the role of the role concept, over which the first role concept has authority, then agent1 has authority over agent2. The *conclude* statement in the rule in figure 8 creates a new belief for the agent executing this rule. This rule is an example of how agents are able to reason with and about defined MOISE$^+$ concepts in the model.

4.2. Behavioral aspect

Next in our framework, we model the behavioral aspect of an organization using the MOISE$^+$ functional specification. A MOISE$^+$ functional specification consists of a number of *schemes* representing a global organizational *goal* in a goal-decomposition tree. A scheme is a joint activity template for the organization, describing which agents should participate

in the scheme. Each goal is assigned to an agent through so-called *missions*. The agent (or agents) that commit to a mission in a scheme commits itself to accomplishing the goals assigned to that mission. A scheme contains a *plan* for executing the scheme root goal and sub-plans for its sub-goals. There are three types of plans; sequence plans, choice plans, and parallel plans.

For agents to reason and behave based on these MOISE+ functional concepts, the agent model needs to include a representation of these concepts, similar to the MOISE+ structural specification described in the previous section. Figure 9 provides the UML class diagram for the MOISE+ functional concepts defined in Brahms.

Fig. 9.   Concepts for modeling MOISE+ functional specification

Having the MOISE+ concepts defined in figure 9 allows us to model a particular scheme for a soccer team. We again use the example from Hübner, et al.[22], see also Section 2.

Fig. 10.   MOISE+ soccer team attack scheme (borrowed from [22])

Figure 10 describes a MOISE+ attack scheme for a soccer team. It describes how three players, performing the scheme jointly, can score a goal. Not shown in figure 10, but stated in [22], each of the three missions (m1, m2, m3) has a cardinality constraint stating they should be committed to by only one player. This constraint defines that three different players have to be committed to the missions to execute the attack scheme. Each mission has a set of goals associated with

it that defines the tasks of each of the three players. Thus, the scheme, together with its missions, describes a *joint activity* between three agents. Besides goals and missions, the scheme in figure 10 provides plans for the agents. Since each agent knows what goals it needs to accomplish based on the mission it has committed to, the tree-structure provides the sequence in which each agent should obtain these goals.

Three missions need to be executed in parallel by three different players, in order to score a goal (see the three sub-goals under the parallel sub-plan in figure 10). At the start of the soccer attack scheme, three players have to agree on the team goal *score a goal*. This is step 1 in the execution of the scheme. How this agreement is to take place is not defined by MOISE+. This is the kind of *work practice* that needs to be developed by the players in the team (c.f. [21]). The next step is for each player to commit to one of the missions. Finally, based on their practice of executing this mission, each player performs the corresponding activities in the correct order. For example, a player that commits to mission *m1* knows that it needs to sequentially perform the sub-goals *get the ball*, *go to the opponent field*, and lastly *kick the ball to (agent committed to m2)*.

The scheme in figure 10 shows that there is a sub-plan with parallel goals, each assigned to an agent with a different mission. In compliance with the cardinality constraint for the three missions, three players perform activities in parallel to obtain their own sub-goals. If the team has practiced this enough, the team members will seamlessly perform this parallel plan. If not, they will fail the overall team aim of scoring a goal.

Figure 11 shows in UML how the soccer attack scheme from figure 10 is implemented in Brahms given the MOISE+ concepts defined in figure 9. Having defined these attack scheme concepts, as shown in figure 11, Brahms agents can get beliefs about these concepts, plus with the declarative knowledge they need to execute the scheme. Figure 12 shows the partial Brahms code for the *MoiseSoccerTeamFunctionalSpecification* object that implements the MOISE+ attack scheme and the UML diagram of figures 10 and 11.

Fig. 11.   Concepts for implementing the MOISE+ soccer team attack scheme, based on the definitions from figure 9

```
object MoiseSoccerTeamFunctionalSpecification instanceof
    MoiseOrganizationStructure
{
  initial_beliefs:
    (current.describesMoiseFunctionalStructureOf = SoccerTeam);
    (SoccerTeamAttackScheme isInstanceOf MoiseSchemeConcept);
    (SoccerTeamAttackScheme hasMoiseMission m1);
    (SoccerTeamAttackScheme hasMoiseMission m2);
    (SoccerTeamAttackScheme hasMoiseMission m3);
    (SoccerTeamAttackScheme.topMoiseGoal = score_a_goal);
    (SoccerTeamAttackScheme hasPlan plan_for_scoring_a_goal);
    (m1 isInstanceOf MoiseMissionConcept);
    (m1 hasMoiseGoal score_a_goal);
    (m1 hasMoiseGoal get_the_ball);
    (m1 hasMoiseGoal go_towards_the_opponent_field);
    (m1 hasMoiseGoal kick_the_ball_to_agent_committed_in_m2);
    (m1.firstMoiseGoalInMission = get_the_ball);
    (m1.numberOfAgentsToCommit = 1);
    (m2 isInstanceOf MoiseMissionConcept);
    (m2 hasMoiseGoal score_a_goal);
    (m2 hasMoiseGoal be_placed_in_the_middle_field);
    (m2 hasMoiseGoal go_to_the_opponent_back_line);
    (m2 hasMoiseGoal kick_the_ball_to_the_goal_area);
    (m2.firstMoiseGoalInMission = be_placed_in_the_middle_field);
    (m2.numberOfAgentsToCommit = 1);
    (m3 isInstanceOf MoiseMissionConcept);
    (m3 hasMoiseGoal score_a_goal);
    (m3 hasMoiseGoal be_placed_in_the_opponent_goal_area);
    (m3 hasMoiseGoal shot_at_the_opponent_goal);
    (m3.firstMoiseGoalInMission =
            be_placed_in_the_opponent_goal_area);
    (m3.numberOfAgentsToCommit = 1);
    (plan_for_scoring_a_goal isInstanceOf MoisePlanConcept);
    (plan_for_scoring_a_goal isPlanForMoiseScheme
            SoccerTeamAttackScheme);
    (plan_for_scoring_a_goal.topMoiseGoal = score_a_goal);
    (plan_for_scoring_a_goal.sequenceMoisePlan(1) =
            get_the_ball);
    (plan_for_scoring_a_goal.sequenceMoisePlan(2) =
            plan_for_field_placement);
    (plan_for_scoring_a_goal.sequenceMoisePlan(3) =
            kick_the_ball_to_agent_committed_in_m2);
    (plan_for_scoring_a_goal.sequenceMoisePlan(4) =
            go_to_the_opponent_back_line);
    (plan_for_scoring_a_goal.sequenceMoisePlan(5) =
            kick_the_ball_to_the_goal_area);
    (plan_for_scoring_a_goal.sequenceMoisePlan(6) =
            shot_at_the_opponent_goal);
    (plan_for_field_placement isInstanceOf MoisePlanConcept);
    (plan_for_field_placement.isMoiseSubPlanFor =
            plan_for_scoring_a_goal);
    (plan_for_field_placement.topMoiseGoal = score_a_goal);
    (plan_for_field_placement.parallelMoisePlan(1) =
            go_towards_the_opponent_field);
    (plan_for_field_placement.parallelMoisePlan(2) =
            be_placed_in_the_middle_field);
    (plan_for_field_placement.parallelMoisePlan(3) =
            be_placed_in_the_opponent_goal_area);
    …
}
```

Fig. 12. Brahms code of MOISE+ soccer team attack scheme from figure 11

4.2.1) Individual Agent Behavior: We illustrate how agents in our example obtain the code they need to exhibit the behavior appropriate for their role. The Brahms language allows us to model the behavior of our agents from the generic MOISE+ behavior, to the domain specific soccer behavior and finally to the individual soccer player agent, depending on the roles these agents play in the organization. In this section we describe how such behavior might be modeled, given the MOISE+ structure we discussed so far. In our example, Ajax coach agent Rinus informs agent Sjakie during the training session about the Ajax team organization. Thus agent Sjakie obtains the beliefs that correspond to the MOISE+ Ajax organization

as presented in Figure 7[3]. After agent Rinus talks to agent Sjakie, he goes and talks to both agent Johan and cm_1. Consequently, agent Johan talks to agent Sjakie about his role in the attack scheme. After this agent Johan tells agent Rinus that he spoke with agent Sjakie. Last, agent Rinus tells all the three agents to go and execute the attack scheme. Figure 13 shows agent Rinus talking to agent Sjakie and thus transferring its beliefs. Together this piece of the simulation shows the work practice performance of the Ajax organization, given our scenario. This agent interaction is not part of the MOISE+ functional structure as discussed in the previous section. The explanation of the joint agent organization performance of the MOISE+ soccer attack scheme comes next.

Figure 14 shows the execution of the soccer attack scheme by the three agents performing mission m1, m2 and m3. Mission m1 is committed to by agent cm_1, m2 by agent Johan and m3 by agent Sjakie. The execution of the scheme starts for each agent, at the same time, with the execution of the workframe *wf_ExecuteMission*. The code for this, and other workframes and activities, can be found in the Appendix. The *ExecuteMission* activity is domain dependent, and is implemented in the domain group *PlayerRole*. This way all soccer player agents will inherit the same activity, and can execute the mission in the same way.

In our example scenario, the only activity that our agents can execute is *PlayingSoccer*. This activity is the main activity that each soccer player agent executes, based on the mission that it is committed to. All sub-activities for playing soccer are defined as workframes inside the *PlayingSoccer* activity (see figure 15). For source code of this activity, see the Appendix.

4.3. Collaboration aspect

The process of collaboration, essential for team work is not defined by MOISE+: a) how the three players on a soccer team decide together to dynamically and instantly create a group or sub-team to execute the attack scheme, b) how the players decide who is committing to which mission, and c) what activities the players should perform to obtain the goals associated with their mission. These collaboration aspects are left to the soccer team, or even more specifically, to the individual players of the soccer team.

In order to model a group of players deciding to manage the performance of a MOISE+ scheme, we adhere to the *team-work model* as described by Sierhuis, et al. in [41]. In this model, the dynamic creation of a group or team of agents follows five distinct phases (see figure 16). Within the soccer team, using the team-work model, the soccer team members can dynamically form sub-teams to fulfill the team's utlitmate goals of scoring more goals than the opposing team.

4.4. Regulation aspect

Regulations in organizations are meant to improve the efficacy of its operations. In human societies norms can be described as the unwritten rules of the organization, in contrast

---

[3]Agent Sjakie_Meulemans gets a lot more beliefs than are shown in Figure 7, because agent Rinus_Michels tells agent Sjakie about all the Ajax team agents, whereas Figure 7 only shows the agents relevant for the example.
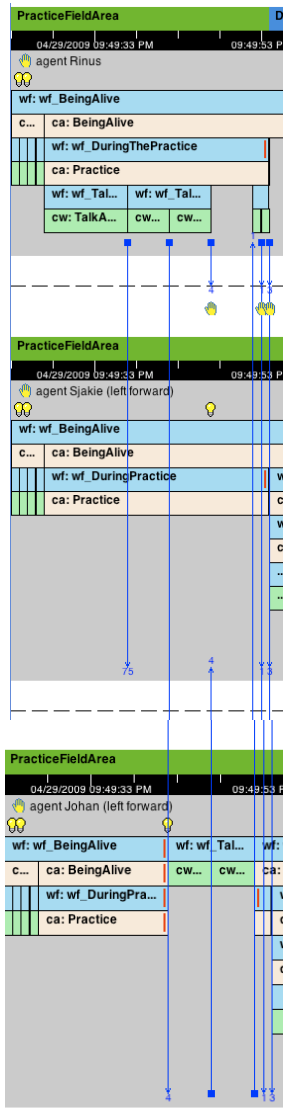
Fig. 13. Brahms sim screenshot of the agent activity timeline for agent Rinus, agent Sjakie, and agent Johan. The vertical (blue) arrows in the screenshot show agent communicatiion over time. The execution of workframes and activities is shown as the (blue, skin and green) colored horizontal bars, while the thoughtframes (reasoning rules) are shown as little lightbulb icons above the activities.The horizontal (green) bar at the top, above the (black) timeline bar, shows the location of the agents (i.e., the PracticeFieldArea). Agent Rinus is first communicating to agent Sjakie during the Practice activiity. Next Rinus is talking to agents Johan and cm_1 (agent cm_1 is not shown, but is shown as a little hand icon underneath agent Rinus). Then you can see agent Johan communicating with agent Sjakie, and the with agent Rinus. All this is being done during the execution of the activity Practice.

with the laws and policies that comprise the written rules. As soon as we simulate an organization, having unwritten rules is impossible. Therefore, unwritten has to be translated to "unofficial." Norms, policies and laws have in common that individuals can decide to violate them. The violation of a norm, if noticed by other agents, can cause the violating agent, for example, to suffer rebukes and/or a loss of reputation. However, violation of a norm cannot be punished by the institution. The violation of policies and laws, if detected, can lead to the same effects as violation of norms, but violation of policies and laws can also be punished by the institution.
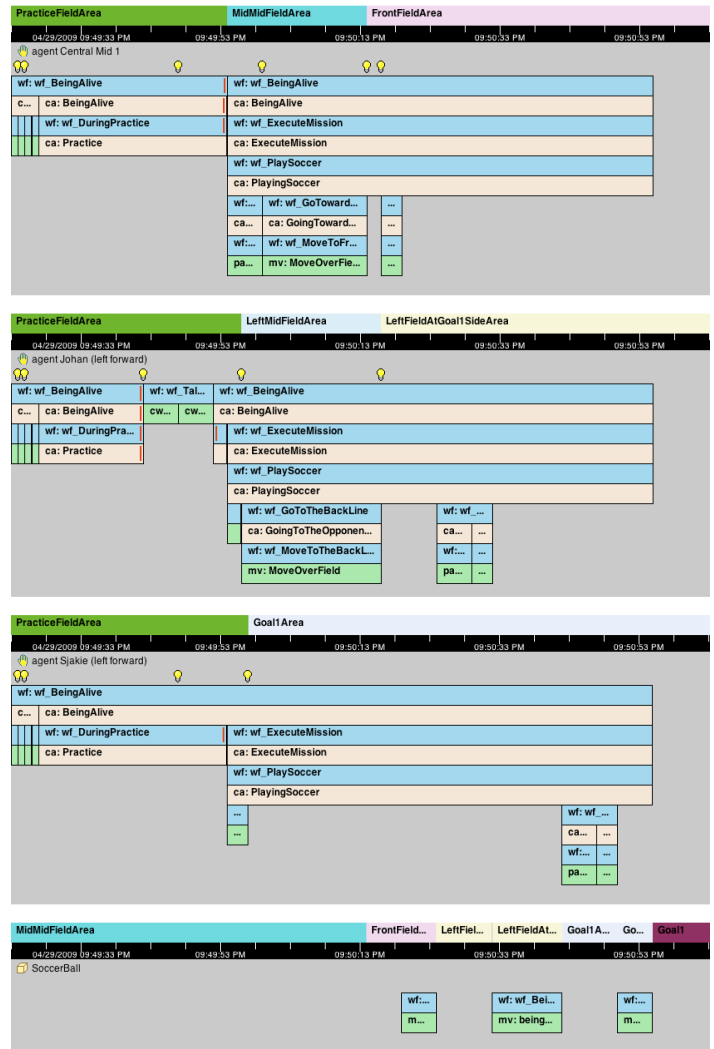

Fig. 14. Brahms sim screenshot of the agent activity timeline for agent Rinus, agent Sjakie, and agent Johan.
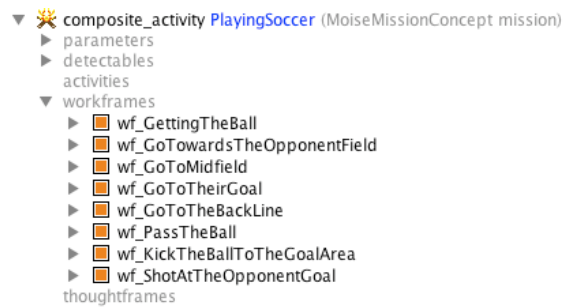

Fig. 15. Brahms sim screenshot of the agent activity timeline for agent Rinus, agent Sjakie, and agent Johan.

Phase Number	Teamwork Phase
1	Recognition of the need of help from other agents
2	Team formation
3	Ongoing coordination and team maintenance throughout task execution
4	Recognition of resolution or impasse
5	Team disbanding

Fig. 16. Teamwork phases to dynamically create and dismember teams

Focus Subject	Individual	Team
Individual	Individual goal for self	Individual goal for the team
Team	Team goal as it applies to the individual	Team goal

Fig. 17.   Four distinct teamwork goals agents have to juggle

As we want to be able to model the full richness of organizations and the agents that populate those organization, we have to be able deal with all possible variations of dealing with norms, policies and laws:

- norms can, but need not be explicitly available in the institution. For example, they can be written in files accessible to all members, but they can also be considered common knowledge and exist in the mental models of the agents that realise the organization. of the organization.
- members of the organization can, but need not be, aware of the norms, policies and laws.
- norms in human organizations are the unofficial rules of the organization. Therefore, enforcement of norms is not done directly, but indirectly by the reactions of others agents to observed norm violation.
- enforcement of policies and laws can be done by constraining the set of executable actions of the agents (e.g. physical restraints of getting on the metro if the agent does not have a ticket), or by monitoring of agent behavior and having ways of punishment if a violation has been detected (e.g. getting a speeding fine if caught).
- With respect to monitoring and punishing, the organization might deploy machinery and/or agents. For example, camera's and police patrols are used to monitor speeding of cars.

The example scenario of Sjakie does not touch upon the regulations in the Ajax team, with the possible exception of the coach Rinus Michels have the authority, apparently accepted by all team members, to determine who plays what role and when. It is not clear from the scenario whether this authority is explicit in the organization specification, or implicit. It could be an emergent property of the current team that can be highly dependent on the characters in the team (including the coach). It could be also a long standing emergent property of the team that is consistent over the slowly, but gradually complete consistency of the team. The last case is an example of a tradition or culture of the team that is reasonable robust against changing membership.

In the implementation of the example in Brahms, we chose to let the agents adopt the regulations in their behavior.

4.5. Common Ground aspect

Upon their first encounter with a previously unknown organization actors need to gain *knowledge* about this organization to be able to successfully participate in it. Knowledge about an organization is needed to understand the opportunities the organization creates for the actor, and to be able to reason whether these opportunities are beneficial to the actor. Moreover, agents may create new organizations (in the broadest sense of the word) and thus share new knowledge amongst the actors within such a newly created organization. That is, we allow for the possibility that agents *self-organize* themselves into a group, team or organization. We first discuss the prerequisites for obtaining knowledge about an existing organization as well as the prerequisites for common ground in creating a new organization, and illustrate the concepts using the soccer team example.

The most important prerequisite in our approach for organization and activity aware agents is that *agents possess the basic concepts to represent organizations, groups, roles, relations, and norms*. These are generally recognized as the most important concepts for modeling organizations and are the building blocks we need to enable agents to reason about organizations [42]. Here we thus assume that *all* agents, as a minimum, have access to the *abstract organization concepts* used in MOISE$^+$. In order to know what a soccer team is, an agent should at least know what it means to be part of a group, to play a role within a group, and to be linked in various ways with other roles. How this is done was discussed in section 4.1. Although in principle it might be possible to learn such concepts, we do not consider this possibility.

In addition to all agents having these basic concepts, at least some of the agents must have knowledge about the structure of the *particular* organization. For example, some agents may have knowledge of what a soccer team is, and, conversely, if no agent that is part of a multi-agent system knows about soccer teams, there is no possibility for those agents to play a role in such a team. Although agents may not have this knowledge themselves, at least one agent they are linked to, or the organization they participate in, must be able to provide this information. This still does not mean, however, that an instance of a soccer team exists; it only means that the conceptual knowledge to represent and reason about such a team is available. Moreover, each agent might have only partial knowledge about the organization they are member of. An agent may have only a partial understanding of the links between roles (authority, communication, compatibility, acquantaince). Although in our soccer example this may be somewhat counterintuitive, an agent might know about the goalkeeper role, but not that this role is part of a soccer team (group).

Besides knowledge of basic organizational concepts such as roles, and concepts about particular organizations such as soccer teams, an agent may also have knowledge about a particular *instance* of an organization. For example, an agent may know that Ajax is a well-known soccer team in The Netherlands. In this paper we assume that anyone playing a role in an existing instance of an organization is aware of the existence of this organization. For example, an agent playing the goalkeeper role in Ajax will be assumed to be aware of the existence of Ajax and it being a member of Ajax.

In the case of Sjakie, we assume that Sjakie knows about and is well aware of all the concepts related to soccer teams and knows that both FC Volendam and Ajax are instances of the soccer team organization structure. Within organizational modeling approaches such as MOISE$^+$, it is assumed that knowledge about the Ajax organization is readily available

to agents that are part of the organization, by means of the organizational "layer" in the system.

In contrast we emphasize the importance of dynamically sharing information that is distributed and not centrally available. To continue the example, in order for Sjakie to participate in the Ajax team, common ground needs to be established between Sjakie and the other members of the Ajax team. For example, there is a need to coordinate the roles that different agents will play. There is a whole range of coordinating mechanisms that are used in organizations, varying from consensus reaching debates, to having an authoritative role dictating how it will be done. For example, assuming it is common knowledge that there can be at most one central forward player, team members might debate and argue about who plays that role. However, as is common in the soccer domain, in our example scenario the team coach decides on these matters and delivers his decision by verbal communication. As such, the coach informs Sjakie and the rest of the team through communication that Sjakie will play the central forward role.

The key capability that is needed to participate in an existing organization is the ability to communicate "partial documents" or *templates* that represent aspects of a particular organization (whether this is an instance or only a conceptual scheme)[4]. In the scenario, Sjakie does not need to be informed about soccer in general, but he needs to be informed of his role, the roles of the other team members and the team regulations, as well as the attack scheme and who commits to the missions in the scheme. In our example, a minimal soccer team template, i.e. an instance of a soccer team document in Brahms, instantiated only with Sjakie, Johan, and cm_1 as the central roles, is communicated to Sjakie. Rinus relies on Johan to provide Sjakie with a template in which all roles and commitments are instantiated with a particular player.

Additional capabilities are needed when agents *self-organize* into a new organization. For example, agents come to the understanding that they share a goal of playing soccer may self-organize themselves temporarily in soccer teams that previously did not exist. A good example are informal neighborhood gatherings where soccer is played. As before, agents need to be aware of the existence of each other to be able to communicate and reach an agreement about the organizational structure. Although MOISE[+] facilitates dynamic creation of groups, it is not as natural for simulation purposes to have an organizational layer available that coordinates such creation. We believe that agents should be able to establish a common ground through *dynamic interaction* about a new created organization. This emphasizes the necessity to coordinate. Instead of a top-down approach, we advocate a bottom-up approach where agents know, for example, which role they are able or want to play, but need to coordinate with other agents to reach a common agreement.

In order to facilitate such dynamic processes of organization creation, we believe it is sufficient to differentiate between different *statuses* of organizations. Apart from an organization that "exists" (an actual instance of a conceptual scheme that is realized by agents playing the roles of the organization) we also allow for an organization that is "under construction." Agents, moreover, need to commit to particular roles within the new organization before a status change from "under construction" to "existing" can be made. Such commitment is established through a process of interaction and "debate" that gives rise to an actual (temporary) organization. This process, again, is facilitated by the exchange of templates (partially instantiated "documents"). Similar arguments can be made for organizations that are in "transition", i.e. regarding the reorganization of organizations as discussed in e.g. [1].

4.6. Environment aspect

The Brahms modeling and simulation framework is strong in modeling and simulating both real time and spatial aspects of the environment, see e.g. [20]. For example, the soccer field on which the agents are "playing soccer" is represented as *areas* part of the geography model. The geography model is a set of hierarchical conceptual *areas* representing soccer field locations (see figure 18). Agents and objects (e.g. the soccer ball) are moving in these location areas. Figure 19 shows the movement of the soccer ball through the soccer field locations, as it is kicked by the agents.
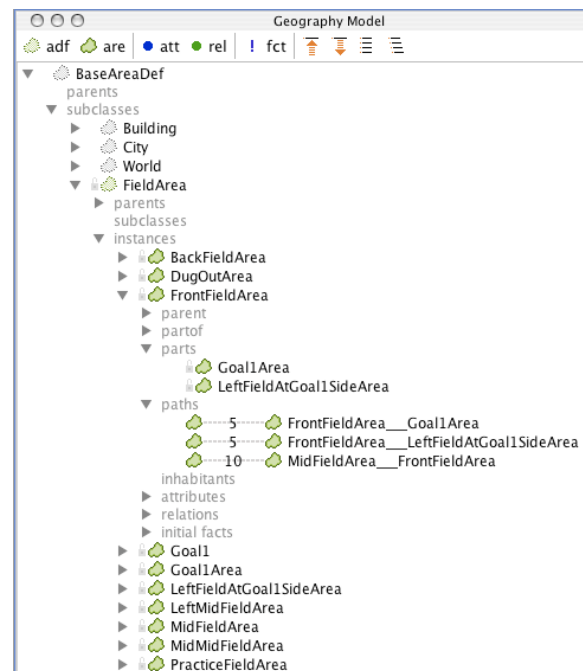

Fig. 18.   Brahms Geography Model for a Soccer Field

Regarding realistic modeling and simulation of organizations, an important aspect is that agents might be incapacitated to fulfill their roles fully or even at all. What is still a matter of research is how to model the consequences of such events. A major issue is to what extent the organization, i.e. other members of the organization know about the problem. Thus the issue of common grounding is closely related to the environment aspect. For example, consider a situation

---

[4]If we would modify our example and Sjakie would not yet be recognized as a talented player, but would be assumed to enter a soccer field for the first time, Sjakie would have to be informed of the soccer conceptual scheme first in order to gain a basic understanding of soccer. For example, Sjakie should be informed that it is not possible to play two different roles at the same time.
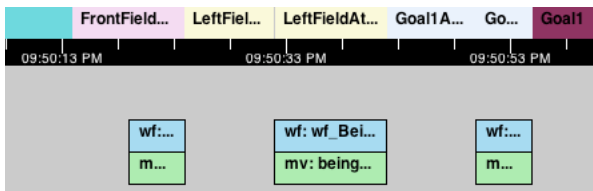
Fig. 19. Movement of the soccer ball through the soccer field areas

that occurs with some regularity at schools: suppose that a teacher on his/her way to school gets involved in an accident, furthermore assume that the nature of the accident disables the communication channels between the teacher and the concierge of the school. As long as nobody knows about the accident and the teacher's involvement, everybody at school (concierge, students, etc.) will assume that the teacher will show up in time for the lectures. The teacher does not show up. Of course this disrupts the normal flow of events, but the problem is more complex from an organizational point of view. Should the role be re-allocated to another agent temporarily or permanently? Does this imply that the teacher is no longer a teacher? Or does it mean that the role of that teacher is at least temporarily assigned to two agents? What if the teacher suddenly does show up at school with the intention of lecturing as usual and finds another teacher in front of the class? More research is necessary to analyse all possible variants of such problems and the corresponding modeling of the organization is concerns.

5. CONCLUSION

The framework for organization modeling and simulation presented in this paper is a first framework that supports organization aware agents. It is characterised by a clear separation of the organization in terms of a structural and behavioral specification, and the agents without which the organization cannot exist. The autonomy of the agents implies that the organization cannot control the way agents reason and try to behave. It can set up environmental constraints that block an agent's actions, but it cannot prevent the agent from deciding to try to take such actions. In the same way, the norms, laws and policies of the organization cannot force an agent to behave in a certain way; they can only entice or convince an agent to behave accordingly as the punishment or loss of reputation enacted upon the agent in case of violation might be prohibitive from the perspective of the agent.

As a consequence, the regulations, and role delegation within the organization cannot just be put into the "head" of the agent that is becoming a member of the organization. The agent has to accept membership, and be informed about the regulations and role delegation. Informing the agent can be done by communication by other members of the organization or by the agent downloading a (partial) specification from a kind of blackboard facility of the organization.

The performance of the organization, therefore, depends entirely on the agents that form it. It depends on the extent to which they adopt the regulations as guidelines for their behavior on the extent to which they are willing to fulfill their

roles, on their capabilities of fulfilling their roles, and on their mutual knowledge and understanding of the way that the other members of the organization fulfill their roles.

An example implementation of the framework has been made in Brahms and based on the organization modeling approach MOISE$^+$. The framework has been instantiated for soccer teams which in turn has been applied to a scenario concerning a new team member.

APPENDIX

This appendix presents partial Brahms source code of the MOISE$^+$ soccer team model. You can download the complete Brahms soccer model from the SoccerModel folder on http://tinyurl.com/brahms-soccer-model. To compile and simulate the model you will need to download and install Brahms from http://www.agentisolutions.com.

```
group MoiseRole memberof MoiseBaseGroup {
...
  composite_activity BeingAlive() {
    end_condition: detectable;
    detectables:
      detectable dt_EndBeingAlive {
        detect ((current.stop = true))
        then end_activity;
      }//dt_EndBeingAlive

    activities:
      composite_activity Practice() {
        end_condition: detectable;

        detectables:
          detectable dt_EndPractice {
            detect ((current.start = true))
            then end_activity;
          }//dt_EndPractice

      }//Practice

    workframes:
      workframe wf_ExecuteMission {
        repeat: false;
        priority: 1;
        variables:
          foreach(MoiseMissionConcept) mymission;
        when (knownval(current.executeMission = mymission) and
          knownval(current.start = true))
        do {
          println_c2("Agent %1 is executing mission: %2",
            current, mymission);
          ExecuteMission(mymission);
        }//do
      }//wf_ExecuteMission
    ...
  }//composite_activity BeingAlive
...
}//MoiseRole
```

Fig. 20. Brahms source code of BeingAlive activity in MoiseRole group.

REFERENCES

[1] A. Lomi and E. Larsen, *Dynamics of Organizations: Computational Modeling and Organization Theories*. Menlo Park: AAAI Press, 2001.

[2] M. Hoogendoorn, C. Jonker, M. Schut, and J. Treur, "Modeling centralized organization of organizational change," *Computational and Mathematical Organization Theory*, vol. 13, pp. 147–184, 2007.

[3] M. Hoogendoorn, C. Jonker, and J. Treur, "Redesign of organizations as a basis for organizational change," in *Coordination, Organizations, Norms and Institutions II, LNAI*, V. Dignum, N. Fornara, P. Noriega, G. Boella, O. Boissier, E. Matson, and J. V. Salceda, Eds. Springer, 2007, vol. 4386, pp. 46–62.

[4] L. Amgoud, N. Maudet, and S. Parsons, "Modelling dialogues using argumentation," in *Fourth International Conference on MultiAgent Systems*. Boston, MA, USA: IEEE, 2000, pp. 31–38.

```
group PlayerRole memberof SocRole {
   …
   workframes:
     workframe wf_BeingAlive {
       detectables:
         detectable dt_MoiseOrganizationStructure {
             detect((<MoiseOrganizationStructure>.
                       describesMoiseOrganizationStructureOf
                       = unknown));
         }//dt_MoiseOrganizationStructure

         detectable dt_FormalOganziation {
             detect((<MoiseOrganizationStructure>.
                       describesFormalOrganizationOf
                       = unknown));
         }//dt_FormalOganziation

         detectable dt_FunctionalStructure {
             detect((<MoiseOrganizationStructure>.
                       describesMoiseFunctionalStructureOf
                       = unknown));
         }//dt_FunctionalStructure

         do {
             Initialize();
             BeingAlive();
         }//do
     }//wf_BeingAlive
}\\PlayerRole
```

Fig. 21.   Brahms source code of the workframe that calls the BeingAlive activity in MoiseRole group.

```
group PlayerRole memberof SocRole {
   …
   activities:
     composite_activity ExecuteMission(MoiseMissionConcept mission) {
       workframes:
         workframe wf_PlaySoccer {
           variables:
             foreach(TeamGroup) team;
             foreach(TeamGroup) subgroup;
           detectables:
             detectable dt_DetectFootballField {
               detect((<FieldArea> isSubAreaOf unknown));
             }
           when (knownval(team.startGame = true)
                   knownval(team hasSubGroup subgroup) and
                   knownval(current isPartOfGroup subgroup))
           do {
             PlayingSoccer(mission);
           }
         }//wf_PlaySoccer
     }//composite_activity ExecuteMission
}\\PlayerRole
```

Fig. 22.   Brahms source code of ExecuteMission activity in PlayerRole group.

```
group PlayerRole memberof SocRole {
   …
activities:
   composite_activity PlayingSoccer(MoiseMissionConcept mission) {
     end_condition: detectable;

     detectables:
       detectable dt_EndOfHalf {
         detect ((TheGame.endOfHalf = true))
         then end_activity;
       }//dt_EndOfHalf

     workframes:
       workframe wf_GettingTheBall {
         when (
           knownval(current.obtainTeamGoalForTheIndividual = true) and
           knownval(current.myIndividualGoalForTheTeam =
                     get_the_ball) and
           knownval(current.location = SoccerBall.location))
         do {
           println_c("Agent %1 is getting the ball", current);
           GettingTheBall();
         }//do
       }//wf_GettingTheBall

       workframe wf_GoTowardsTheOpponentField {
         when (
           knownval(current.obtainTeamGoalForTheIndividual = true) and
           knownval(current.myIndividualGoalForTheTeam =
                     go_towards_the_opponent_field))
         do {
           println_c("Agent %1 is going towards the opponent's goal",
                       current);
           GoingTowardsTheOpponentField();
         }//do
       }//wf_GoTowardsTheOpponentField

       workframe wf_GoToMidfield {
         variables:
           forone(MoiseGoalConcept) nextGoal;
         when (
           knownval(current.obtainTeamGoalForTheIndividual = true) and
           knownval(current.myIndividualGoalForTheTeam =
                     be_placed_in_the_middle_field) and
           knownval(be_placed_in_the_middle_field.
                     nextMoiseGoalInSequence = nextGoal))
         do
           println_c("Agent %1 is going to be placed in midfield",
                       current);
           GoingToMidfield();
           MoveOverField(Arena, LeftMidFieldArea, "GoingToLeftMidfield");
           conclude((current.myTeamGoalForTheIndividual = nextGoal));
           conclude((current.myIndividualGoalForTheTeam = nextGoal));
           conclude((current.obtainTeamGoalForTheIndividual = true));
         }//do
       }//wf_GoToMidfield
   }//composite_activity PlayingSoccer
}//group PlayerRole
```

Fig. 23.   Brahms source code of PlayingSoccer activity in PlayerRole group.

[5] P. McBurney, S. Parsons, and M. Wooldridge, "Desiderata for agent argumentation protocols," in *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*.   New York, NY, USA: ACM, 2002, pp. 402–409.

[6] K. Atkinson, T. Bench-Capon, and P. Mcburney, "A dialogue game protocol for multi-agent argument over proposals for action," *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 2, pp. 153–171, 2005.

[7] T. v. d. Weide, F. Dignum, J.-J. C. Meyer, H. Prakken, and G. Vreeswijk, "Personality-based practical reasoning," University of Utrecht, Tech. Rep., 2007.

[8] M. Tambe, "Towards flexible teamwork," *Journal of Artificial Intelligence Research*, vol. 7, pp. 83–124, 1997.

[9] D. N. Pynadath and M. Tambe, "The communicative multiagent team decision problem: Analyzing teamwork theories and models," *Journal of Artificial Intelligence Research*, vol. 16, pp. 389–423, 2002.

[10] X. Fan and J. Yen, "Modeling and simulating human teamwork behaviors using intelligent agents," The Pennsylvania State University, School of Information Sciences and Technology, Tech. Rep., 2004.

[11] X. Fan, B. Sun, S. Sun, M. McNeese, and J. Yen, "Rpd-enabled agents teaming with humans for multi-context decision makingmulti-c," in *AAMAS 2006*.   Hakodate, Hokkaido, Japan: ACM, 2006.

[12] F. Trompenaars, *Riding the waves of culture*.   London: The Economists Books, 1993.

[13] G. Hofstede, *Cultures Consequence, 2nd ed.*   Thousand Oaks, CA: Sage Publications, 2001.

[14] V. Subrahmanian, M. Albanese, M. V. Martinez, D. Nau, D. Reforgiato, G. I. Simari, A. Sliva, O. Udrea, and J. Wilkenfeld, "Cara: A cultural-reasoning architecture," *IEEE Intelligent Systems*, vol. 1541, no. 1672, pp. 12–15, 2007.

[15] L. Yilmaz and T. I. Ören, *Agent-Directed Simulation and Systems Engineering.*, ser. Wiley Series in Systems Engineering and Management. Berlin, Germany: Wiley, In Press.

[16] W. Clancey, M. Sierhuis, B. Damer, and B. Brodsky, "The cognitive modeling of day in the life social behaviors using brahms," in *Cognition and Multi-Agent Interaction*, R. Sun, Ed.   New York, NY: Cambridge University Press, 2005, pp. pp. 151–184.

[17] W. J. Clancey, "Simulating activities: Relating motives, deliberation, and attentive coordination," *Cognitive Systems Research*, vol. 3, no. 3, pp. 471–499, 2002.

[18] G. Klein, P. Feltovich, J. Bradshaw, and D. D. Woods, "Common ground and coordination in joint activity," in *Organizational simulation*, W. Rouse and K. Boff, Eds.   New York: Wiley-Interscience, 2005, pp. 139–184.

[19] A. Omicini, A. Ricci, M. Viroli, C. Castelfranchi, and L. Tummolini, "Coordination artifacts:environment-based coordination for intelligent agents," in *Third International Joint Conference on Autonomous Agents*

and Multiagent Systems, vol. 1. New York, NY: IEEE Computer Society, 2004.

[20] M. Sierhuis, W. J. Clancey, and R. J. v. Hoof, "Brahms: An agent-oriented language for work practice simulation and multi-agent systems development," in *Multi-Agent Programming, 2nd Edition*, R. H. Bordini, M. Dastani, J. Dix, and A. E. Fallah-Seghrouchni, Eds. Springer, In Press.

[21] W. Clancey, P. Sachs, M. Sierhuis, and R. v. Hoof, "Brahms: Simulating practice for work systems design," *International Journal on Human-Computer Studies*, vol. 49, pp. 831–865, 1998.

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

[23] J. Gratch and S. Marsella, "A domain-independent framework for modeling emotion," *Cognitive Systems Research*, vol. 5, no. 4, pp. 269–306, 2004.

[24] 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)*, ser. LNCS, vol. 4457. Springer, 2007, pp. 86–105.

[25] M. Prietula, K. Carley, and L. Gasser, *Simulating Organizations: Computational Models Of Institutions and Groups*. Menlo Park, CA: AAAI/MIT Press, 1998.

[26] S. Moss, H. Gaylard, S. Wallis, and B. Edmonds, "Sdml: A multi-agent language for organizational modelling," *Computational and Mathematical Organization Theory*, vol. 4, no. 1, pp. 43–70, 1998.

[27] 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)*, ser. LNCS, vol. 2935. Springer, 2003, pp. 214–230.

[28] F. Brazier, P. van Eck, and J. Treur, "Modelling a society of simple agents: From conceptual specification to experimentation," *Journal of Applied Intelligence*, vol. 14, pp. 161–178, 2001.

[29] M. Esteva, J. Padget, and C. Sierra, "Formalizing a language for institutions and norms," ser. LNCS, J.-J. C. Meyer and M. Tambe, Eds., vol. 2333. Springer, 2002, pp. 348–366.

[30] J. Vazquez-Salceda, *The Role of Norms and Electronic Institutions in Multi-Agent Systems: The HARMONIA Framework*, ser. Whitestein Series in Software Agent Technologies and Autonomic Computing. Birkhäuser, 2004.

[31] C. Jonker, A. Sharpanskykh, J. Treur, and P. Yolum, "A framework for formal modeling and analysis of organizations," *Journal of Applied Intelligence*, vol. 27, pp. 49–66, 2007.

[32] V. Dignum, Ed., *Multi-Agent Systems: Semantics and Dynamics of Organizational Models*. IGI Global, 2009, to appear.

[33] F. Dignum, V. Dignum, and C. Jonker, "Towards agents for policy making," 2008.

[34] B.-J. v. Putten, V. Dignum, M. Sierhuis, and S. R. Wolfe, "Opera and brahms: a symphony?" in *Agent-Oriented Software Engineering (AOSE) 2008 at The Sixth International Joint Conference on Autonomous Agents & Multi-Agent Systems (AAMAS 2008)*, vol. Forthcoming LNCS Proceedings. Estoril, Portugal: Springer, 2008.

[35] 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)*. IEEE Computer Society, 2004, pp. 236–243.

[36] 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)*. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 1677–1678.

[37] J. Bradshaw, M.Sierhuis, A. Acquist, P. Feltovich, R. Hoffman, R. Jeffers, D. Prescott, N. Suri, A. Uszok, and R. v. Hoof, "Adjustable autonomy and human-agent teamwork in practice: An interim report on space applications," in *Agent Autonomy*, H. Hexmoor, C. Castelfranchi, and R. Flacone, Eds. Kluwer, 2003.

[38] A. Uszok, J. M. Bradshaw, M. Johnson, R. Jeffers, A. Tate, J. Dalton, and S. Aitken, "Kaos policy management for semantic web services," *IEEE Intelligent Systems*, vol. 19, no. 4, pp. 32–41, 2004.

[39] C. Castelfranchi, F. Dignum, C. M. Jonker, and J. Treur, "Deliberative normative agents: Principles and architecture," in *Intelligent Agents VI. Agent Theories Architectures, and Languages, ATAL99*, N. R. Jennings and Y. Lesperance, Eds. Springer Berlin / Heidelberg, 2000, vol. 1757, pp. 364–378.

[40] F. Dignum, B. Edmonds, and L. Sonenberg, "The use of logic in agent-based social simulation," *Journal of Artificial Societies and Social Simulation*, vol. 7, no. 4, 2004.

[41] M. Sierhuis, J. M. Bradshaw, A. Acquisti, R. v. Hoof, R. Jeffers, and A. Uszok, "Human-agent teamwork and adjustable autonomy in practice," in *The 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, Nara, Japan, 2003.

[42] E. Bottazzi and R. Ferrario, "A path to an ontology of organizations," in *Proceedings of International EDOC Workshop on Vocabularies, Ontologies and Rules for The Enterprise (VORTE 2005)*, 2005, pp. 9–16.

Maarten Sierhuis Maarten Sierhuis is senior scientist at NASA Ames Research Center (ARC) and adjunct professor at Carnegie Mellon Universitys Silicon Valley campus. He is also a visiting professor in the Man-Machine Interaction group at Delft University of Technology. He is co-principal investigator of the Brahms agent-oriented language and simulation environment, in the Work Systems Design & Evaluation group in the Collaborative and Assistant Systems area within the Intelligent Systems Division at NASA ARC. He holds a PhD in social science informatics from the University of Amsterdam, in which he developed a theory for modeling and simulating work practice using Brahms. Prior to joining NASA, Sierhuis was a member of technical staff at NYNEX Science & Technology (1990-1997); earlier he was at IBM in New York and Sema Group in The Netherlands (1986-1990). He has presented invited lectures on Brahms, has published widely in this area and holds two software patents involving work practice simulation and hypertext databases. For more information visit: http://homepage.mac.com/msierhuis

Catholijn Jonker Catholijn Jonker is full professor of Man-Machine Interaction at the Faculty of Electrical Engineering, Mathematics and Computer Science of the Delft University of Technology. She studied computer science, and did her PhD studies at Utrecht University. After a post-doc position in Bern, Switzerland, she became assistant (later associate) professor at the Department of Artificial Intelligence of the Vrije Universiteit Amsterdam. From september 2004 unitl september 2006 she was a full professor of Artificial Intelligence / Cognitive Science at the Nijmegen Institute of Cognition and Information of the Radboud University Nijmegen. She chaired De Jonge Akademie (Young Academy) of the KNAW (The Royal Netherlands Society of Arts and Sciences) in 2005 and 2006, and she is a member of the same organization from 2005 through 2010.

Birna van Riemsdijk M. Birna van Riemsdijk is assistant professor in the Man-Machine Interaction group at Delft University of Technology., The Netherlands. Until september 2008, she was a postdoc in the Programming and Software Engineering group at Ludwig-Maximilians-University in Munich, and she obtained here PhD in the Intelligent Systems group at Utrecht University, The Netherlands. She has done research in the areas of agent programming and service-oriented systems. She is a member of the steering committee of the International Workshop on Declarative Agent Languages and Technologies (DALT), and has been a co-chair of this workshop several times. She is a PC member for numerous international workshops and conferences in the field of agent technology and intelligent systems.

Koen Hindriks Koen Hindriks is Assistant Professor at the Man-Machine Interaction group at the Faculty of Electrical Engineering, Mathematics and Computer Science of the Delft University of Technology. He studied computing science, and finished his PhD at Utrecht University on agent programming languages. His research interests include common-sense reasoning, agent-oriented programming based on common sense concepts like beliefs and goals, and the verification and specification of agent programs. He has designed and developed several agent programming languages, including 3APL and GOAL. He is also interested in the design and development of negotiating agents, which involves among others research on representation, strategies and learning techniques that can be usefully applied in the development of such agents.