Künstl Intell (2012) 26:37–45
DOI 10.1007/s13218-011-0152-5

FACHBEITRAG

Agreeing on Role Adoption in Open Organisations

Huib Aldewereld · Virginia Dignum · Catholijn M. Jonker · M. Birna van Riemsdijk

Received: 28 July 2011 / Accepted: 4 November 2011 / Published online: 24 November 2011

Abstract The organisational specification of a multi-agent system supports agents' effectiveness in attaining their purpose, or prevent certain undesired behaviour from occurring. This requires that agents are able to find out about the organisational purpose and description and decide on its appropriateness for their own objectives. Organisational modeling languages are used to specify an agent system in terms of its roles, organizational structure, norms, etc. Agents take part in organisations by playing one or more of the specified roles for which they have the necessary capabilities.

In this paper, we investigate the process of role adoption in the context of the well-known OperA organisational modelling language. In OperA, each organisation has a gatekeeper role responsible for admitting agents to the organisation. Agents playing the role of gatekeeper can interact with agents that want to enter the organisation in order to come to agreement on role adoption. That is, negotiate which roles they will play and under which conditions they will play them. This is possible by evaluating capability requirements for roles. We extend OperA to allow for the specification of role capabilities. This approach will be illustrated using the Blocks World for Teams (BW4T) domain.

H. Aldewereld · V. Dignum (✉) · C.M. Jonker · M.B. van Riemsdijk
Delft University of Technology, Delft, The Netherlands
e-mail: m.v.dignum@tudelft.nl

H. Aldewereld
e-mail: h.m.aldewereld@tudelft.nl

C.M. Jonker
e-mail: c.m.jonker@tudelft.nl

M.B. van Riemsdijk
e-mail: m.b.vanriemsdijk@tudelft.nl

Keywords Agent programming · Organizational modeling · Role enactment

1 Introduction

Agent organisations have been motivated as a suitable way to deal with agent autonomy, especially in open multi-agent systems (MAS) [10], where in principle any agent can enter into the system. In open systems, agents, representing people, groups or enterprises, are assumed to be developed and owned by stakeholders other than the owners of the organisation model. For example, consider an agent that is to help an user locate a best deal for a product on the Web. The agent is to access web stores, compare products and prices, negotiate with the web stores and reach a good agreement for the product at one of these web stores. Given that the web stores are designed by other developers than this agent, the agent must be able to understand the structure and procedures of the webstore in order to play its role within the webstore (i.e. depending on the webstore structure, this role may be that of visitor, searcher or buyer). Information exchange between sources belonging to different organisations will become the standard. Similarly, ad hoc organisations will need to be formed to satisfy emerging information needs.

The development of such open organisation, must enable the validation of its results even though the organisation's behaviour is dependent on the behaviour of the external participating agents. That is, system design must consider not only the organisational characteristics such as stability over time, some level of predictability, and commitment to aims and strategies, etc. but also the individual autonomy of the participants. This requires richer organisational concepts than those provided by most MAS models.

The concept of agent organisation, supporting the specification of roles, interaction structures, and norms (see, e.g., [8, 13]), solves the inherent issues of coordination in open agent systems. Agent organisation distinguishes between the mechanisms through which the structure and global behaviour of the model is described and coordinated, and the agents that populate the model enabling its animation. Therefore, organisation frameworks should represent interaction in a way that (1) is independent of the internal design of the agents, and (2) is able to integrate organisational characteristics and demands with the agent's own goals in a dynamic way that preserves the autonomy of the participating agents. Contracts between agent and organisation enable the flexible instantiation of roles, to conjugate the top-down specification of organisational structures with the autonomy of participating agents. This abstraction, however, leads to two separate problems.

Firstly, agents who want to enter and play roles in an organisation are expected to understand and reason about the organisational specification, if they are to operate effectively and flexibly in the organisation. In earlier work we have looked at such agents capable of this organisational reasoning [23]. An important aspect of these *organisation-aware agents* is the ability to reason about role enactment. In particular, an agent has to reason about whether it wants to play a role and whether it has the capabilities to behave as the role requires. In [24] we discuss reflection on role enactment and propose an interaction protocol through which agents can apply for enacting roles in the organisation.

Secondly, there is the issue of selecting suitable agents to play a role in the organisation successfully. A possible way to ensure this is by introducing a dedicated agent (a *gatekeeper*) that is responsible for admitting agents to the organisation. An example of an organisational modelling language in which such a gatekeeper is present, is OperA [8]. The idea of the interaction between the agent and the organisation is that the gatekeeper asks agents who want to join whether they have the necessary capabilities for playing the desired role in the organisation, and then assigns roles to agents based on this. In earlier work [24], we have proposed a protocol for this interaction, but it was left unclear how the gatekeeper decides on the role assignments. In this paper we focus on the reasoning aspects of the gatekeeper agent. How to decide which agent to choose for fulfilling (critical) roles? And, can any agent that says that it has the required capabilities be trusted? This approach requires that necessary capabilities for enacting a role can be described in the role specification.

This paper proposes a framework to deal with these two problems. The framework consists of an interaction protocol for agents that want to enter open organisations, The framework is formalised as an extension to the OperA model. Of course formalisations to other organisation modelling languages can be defined in a similar way. The ideas and formalisations have been tested in the Blocks World for Teams environment (BW4T) [14].

The paper is organised as follows. In Sect. 2, we describe Blocks World for Teams, the teamwork domain used as illustration scenario to investigate the reasoning process of the gatekeeper on agent admission. Section 3 presents a general pattern for modeling capabilities in OperA, which are an essential piece of information for the gatekeeper agent. We propose a negotiation and agreement process for agent admission in Sect. 4. We conclude the paper and discuss future research in Sect. 5.

2 Blocks World for Teams

The Blocks World for Teams (BW4T) simulated environment [14] has been developed as a testbed for human-agent/robot teamwork. The environment consists of a number of rooms that are connected through halls. Coloured blocks are placed inside the rooms. Simulated robots should work together to pick up blocks from the rooms, bring them to the so-called drop zone and put them down there, in a specified colour sequence. Blocks only become visible once a robot enters the room where these blocks are. Robots cannot see each other but they can exchange messages. Once a robot enters a room (including the drop zone), no other robots can enter. Blocks disappear from the environment when dropped in the hall or in the drop zone. Robots can be controlled by agents or humans, thereby providing the possibility to investigate human-agent robot teamwork. In this paper, we only consider agent-only teams.

An interface that allows a GOAL [11] agent to control a simulated robot has been developed using the Environment Interface Standard (EIS) [2]. Broadly speaking, this standard specifies that agents can control entities in the environment through actions, and that agents can observe the environment through percepts that are sent from the environment to the agents. The actions made available to agents in the BW4T environment are `goTo(<Place>)` to move to the specified place (a room, the drop zone or a hall), `goTo(<Block>)` to move to the specified block, `pickUp` to pick up a block (the robot has to be close to the block) and `putDown` to put a block down (if the robot is not holding a block, the action has no effect).

Percepts available to agents include `at(<Me>, <Place>)` which specifies the current location of the robot, and `colour(<Block>,<Colour>)` which is sent when an agent enters the room where `<Block>` is located.

The colour sequence in which agents should put down blocks at the drop zone is sent at the beginning to all agents using percept `sequence([<Colour>])` which has a list

of colours as parameter. The colour to deliver is indicated to agents using the percept `sequenceIndex(<N>)`, where `<N>` refers to the N-th element in the colour sequence.

Designing an open organisation for the deceptively simple looking BW4T presents the designer with all the important challenges we want to address with our framework. For example, the environment has *limited visibility*; *actions take time*, which means that effective cooperation, in order to prevent unnecessary actions, can significantly reduce the time needed to deliver all the required blocks; and *communication* is needed that can support effective teamwork, for example by letting other agents know which blocks one has discovered and which block one is carrying. A dependency analysis of the BW4Ts shows its true complexity [15].

3 Organisational Specification

The OperA framework [8] proposes an expressive way for defining open organisations distinguishing explicitly between the organisational aims and the agents who act in it. That is, OperA enables the specification of organisational structures, requirements and objectives independently from any knowledge on the properties or architecture of agents, which allows participating agents to have the freedom to act according to their own capabilities and demands. The OperA framework consists of three interrelated models: organisation, social, and interaction.

The *Organisational Model* (OM) is the result of the observation and analysis of the domain and describes the desired behaviour of the organisation, as determined by the organisational stakeholders in terms of roles, objectives, norms, interactions and ontologies. The design and validation of OperA OMs can be done with the OperettA tool [1]. The OM provides the overall organisation design that fulfills the stakeholders' requirements. The OM consists of a *social structure* which describes roles and their objectives, and relations between roles concerning achievement of objectives (further detailed in Sect. 3.1), a *normative structure* which describes norms associated with roles (further detailed in Sect. 3.2), an *interaction structure* which is an abstract workflow that specifies how objectives should be achieved by the organisation using the notions of scenes and landmarks, and a *communicative structure* which specifies the organisation's ontology and communicative languages. In this paper, for simplicity, we assume a common ontology for the organisation and the agents. The OM is a descriptive view of the organisation. In itself, the OM cannot act but is dependent on a population of agents that enact its roles in order to achieve the organisation's objectives. What this population looks like and how it acts are described in OperA in the Social and Interaction Models.

The *Social Model* (SM) maps roles to agents and describes agreements concerning the role enactment and their conditions in social contracts. In OperA, agent and role are fundamentally different concepts. Roles are typically declarative entities meant to represent a part of the organisation's design and can be taken up by the agents enacting the role. Objectives of an organisation are achieved through the actions of agents, which means that, at each moment, an organisation should employ the relevant agents that can make its objectives happen. That is, a role only gets an operational semantics indirectly through the agents that take up that role. For the operationalisation of OperA organisations, a *Gatekeeper* role is defined, which is responsible for the assignment of roles to (external) agents. The gatekeeper agent is part of the SM of each organisation.

Finally, the *Interaction Model* (IM) specifies the interaction agreements between role-enacting agents as interaction contracts. The IM specification enables variations to the enactment of interactions between role-enacting agents.

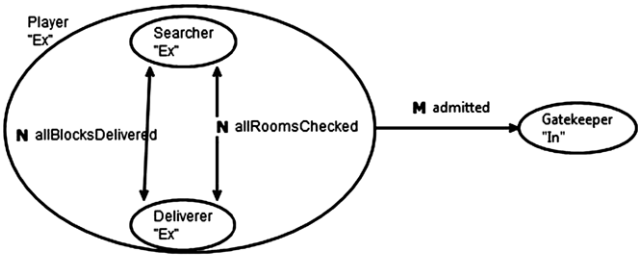
In this paper, we focus on the SM, specifically on the role assignment processes. Therefore, only those components of the OM that are relevant for this aspect are considered, in particular the social and the normative structures which are described in more detail in the remainder of this section. Moreover, in order to enable negotiation on role enactment, we extend OperA with the definition of role capabilities, as discussed in Sect. 3.3.

3.1 Social Structure

The *Social Structure* of an organisation describes the roles in the organisation. For example, who should locate blocks, who should pickup and return them, how they relate to each other, etc. The Social Structure consists of a list of role definitions, *Roles* (including their objectives, rights, norms and requirements); a list of role groups' definitions, *Groups*; and a *Role Dependencies* graph.

Abstract organisation objectives form the basis for the definition of the objectives of roles. From the organisation perspective, role descriptions identify the activities and services necessary to achieve its objectives and enable to abstract from the individuals that will eventually perform the role. From the agent perspective, roles specify the expectations of the society with respect to the agent's activity in the society. In OperA, the definition of a role consists of an identifier, and sets of role objectives (with possibly their sets of sub-objectives), role rights, norms, and the role type.

Figure 1 shows the social structure of the BW4T organisation as used in this paper, and the corresponding role descriptions for the *Searcher* and *Deliverer* roles. Player is a group containing the roles Searcher and Deliverer. The arcs in the social structure diagram indicate the dependency relations between roles or groups. That is, the arcs are labeled with the objectives for which the parent role depends on the child role. These dependencies describe how the distribution



Property	Value
Concept Name	Concept Searcher
Dependant On	N Network Dependency allBlocksDelivered, I
Dependee In	N Network Dependency allBlocksDelivered, I
Name	Searcher
Norms	Norm capN1: able to go to a place, Norm
Objectives	Objective allRoomsChecked
Role Type	Ext

Property	Value
Concept Name	Concept Deliverer
Dependant On	N Network Dependency allBlocksDelivered,
Dependee In	N Network Dependency allBlocksDelivered,
Name	Deliverer
Norms	Norm capN6: able to go to a place, Norm
Objectives	Objective allBlocksDelivered
Role Type	Ext

Fig. 1 Role dependencies (*top*), properties of Searcher (*middle*) and Deliverer (*bottom*)

of objectives in the organisation is realised. OperA identifies three types of role dependencies: bidding [**M**arket], request [**N**etwork], and delegation [**H**ierarchy]. These are important for the realisation of the interactions between role enacting agents within the scenes described in the interaction structure, which is outside the scope of this paper.

In the BW4T example, the organisational objective of collecting the coloured blocks in a particular colour order is split over the two roles in the organisation; the *Searcher*s are responsible for checking all rooms for the blocks and providing the information about block locations and colours to other agents (*allRoomsChecked*), and the *Deliverer*s are responsible for picking up the blocks of the correct colour and dropping them at the drop zone (*allBlocksDelivered*). The deliverers thus depend on the searchers for finding the correct blocks, and the searchers depend on the deliverers for collecting the blocks and bringing them to the drop zone.

The *Gatekeeper* role is not specific to the BW4T domain, but must be present in every OperA organisational model. The gatekeeper is responsible for admitting agents to the organisation by means of *asking agents* about their capabilities and *assigning roles* to agents on the basis of this. This is why the Gatekeeper role has been marked as internal ("In") in the social structure, which means that the agent(s) enacting this role are to be programmed by the designer of the organisation herself, while the other roles are marked as external ("Ex").

External roles can be played by agents that are designed independently from the society. Individual agents consider joining an organisation when they believe that the enactment of role(s) will contribute to the achievement of some of their own goals. When an agent applies, and is accepted for a role, it commits itself to the realisation of the role's objectives and it should function within the society according to the constraints applicable to its role(s). This means that agents need to be able to interpret the specification of the role and take this into account in their decision making. These processes are specified in the interaction structure. The social contracts in the Social Model are the result of these processes.

3.2 Normative Structure

At the highest level of abstraction, norms are the *values* of a society, in the sense that they define the concepts that are used to determine the value or utility of situations. However, values do not specify *how*, *when* or in *which* conditions individuals should behave appropriately in any given social setup. In OperA, these aspects are defined in the *Normative Structure* using a deontic logic that is temporal, relativised (in terms of roles and groups) and conditional. The normative structure enables the definition of norms that specify desired behaviour that agents should exhibit when playing the role.

Norms in OperA can be obligations, permissions or prohibitions, represented respectively as $O_r\varphi$, $P_r\varphi$ and $F_r\varphi$, where r is a role and φ a predicate in the domain language. Examples of norms in the BW4T domain are the obligation for deliverers to inform others of the blocks that they placed in the drop zone, and the prohibition that more than one searcher is present in the same room at any given moment. The latter can be formalised as:

$$F_{searcher}(enter(\langle Room\rangle)|occupied(\langle Room\rangle))$$

3.3 Capabilities in OperA

In the theory of agency, the notions of agent capability and action have been widely researched. The intuition is that an agent possesses capabilities that make action possible. In the literature, there are many approaches to the formalisation of these definitions.[1] Research on the theory of action follows two main schools [9]. The first aims at the explicit representation of action by a specific agent, in terms of dynamic logic or situation calculus; whereas the second is concerned with representing the fact that a certain result has been achieved, such as in the *stit* theories or in the notion of agency. Based on the philosophical notion that "can" implies *ability* and *opportunity* [7], in [9], we have defined agent capability as the inherent skills an agent is endowed with. This implies that having a capability is a necessary but not sufficient prerequisite for exhibiting behaviour that demonstrates the capability. In that work, we discussed the relation between

[1] A concise overview can be found in [6].

agent capability, ability (the potential or opportunity to act in a certain state) and action (the actual act of bringing a certain state about). This interpretation of capability as ability follows [19], in which the notion of capability is investigated in the context of BDI logic. In agent programming, capabilities have been introduced as a modularisation construct [4, 5] comparable to modules in GOAL. Capabilities allow for packaging a subset of beliefs, plans, and goals into an agent module and to reuse this module wherever needed. That is, capability is taken as the capability to achieve goals, which is also the capability type considered in [19].

From an organisation perspective, it is necessary not only to know an agent's capabilities, but also how to match those capabilities to the requirements of the organisation. Instead of agent capabilities, organisation models should describe *role capabilities*, that is, the capabilities that an agent must possess in order to be able to enact that role.

Although capabilities have been considered in the context of the gatekeeper in [8], modeling capabilities has so far received relatively little attention in OperA. Taking into account the analysis of the BW4T domain and other scenarios, we distinguish four *capabilities types* [24]: capabilities to execute *actions*, to *perceive* aspects of the environment in which the agents operate, to *communicate* information, questions or requests, and to achieve *goals* (formulated as objectives in the organisational specification). We believe this to be a suitable distinction since these four types of capabilities correspond to the commonly adopted notion of intelligent agents as being reactive (able to perceive and react to changes in the environment), proactive (act towards achieving goals) and social (communicate with other agents).

We have extended the meta-model of OperA to include the definition of role capabilities as follows:[2]

Definition 1 (Role Capability) Given a role r, a role capability is a predicate describing a requirement for role enactment. A role capability is represented by $cap_r(type, p)$, where $type \in \{(ableToDo, ableToPerceive, ableToSay, ableToAchieve)\}$ and p is a predicate in the domain language.

The semantics of role capabilities is defined as deontic expressions. That is, capabilities can be defined as norms that apply to agents that enact a role. Capability norms describe the activation and expiration conditions related to the enactment and deactment of the role by an agent. Required capabilities are modelled as maintenance conditions, i.e., as long as the agent is playing the role, it is obliged to have that capability. Formally:

- $cap_r(ableToDo, \alpha) \equiv O_r(ableToDo(\alpha))$
- $cap_r(ableToPerceive, p) \equiv O_r(ableToPerceive(p))$
- $cap_r(ableToSay, c) \equiv O_r(ableToSay(c))$
- $cap_r(ableToAchieve, \phi) \equiv O_r(ableToAchieve(\phi))$

where α is an action that can be executed by agents in the environment, p is a situation that can be perceived by agents in the environment, c is a communicative act that can be performed by agents, and ϕ expresses a property of the environment that agents should be able to achieve.

In the BW4T scenario, examples of capabilities for the searcher role are the capability to execute the action of going to a place, the capability to perceive blocks and their colours, the capability to send information about blocks to other agents, and the capability to achieve the goal of having checked all rooms. For example, the required capability of the *Searcher* to perceive the blocks' colours is modelled as

$$O_{Searcher}(ableToPerceive(color(\langle BlockId\rangle, \langle ColorId\rangle)))$$

I.e., upon enacting the role *Searcher*, the agent is obliged to be able to perceive the colour of the blocks until it deacts the role.

Intuitively, role capabilities describe the minimum skills that an agent must possess in order to enact the role. In that sense capabilities are complementary to role rights, which are capabilities that an agent receives *by* enacting a role. Which and how many capabilities are to be required from agents enacting a role is a modeling choice and reflect the balance between regulation and autonomy demanded by the application [20]. Basically, the less demands are placed on role enactment, the more autonomy is allowed to specific role-enacting agents, but the less guarantees can be made with respect to the overall organisational behaviour.

4 Agreeing on Role Enactment

Given heterogeneous and autonomous agents, which have their own goals and interests, mismatches between the goals and capabilities of the agent and the objectives and requirements of a role must be assumed. Thus, participants are admitted to the society only through a process of socialisation, during which the participant negotiates with the organisation (represented by the Gatekeeper) the terms and conditions of its participation. The Social Model (SM) of OperA enables the specification of terms and conditions for participation as a social contract. The *Gatekeeper* role, and the interaction scene *Agent-Admission* are modelled in the Organisational Model.

The Gatekeeper is responsible for admitting agents into the organisation and giving them the role specifications. It has the responsibility of checking the capabilities of the agents and seeing if they match the requirements set for the role. As such, the Gatekeeper plays an important function in the SM. In previous work we have introduced a protocol for the interaction between an agent wanting to join

[2] See [8] for the full specification of OperA.

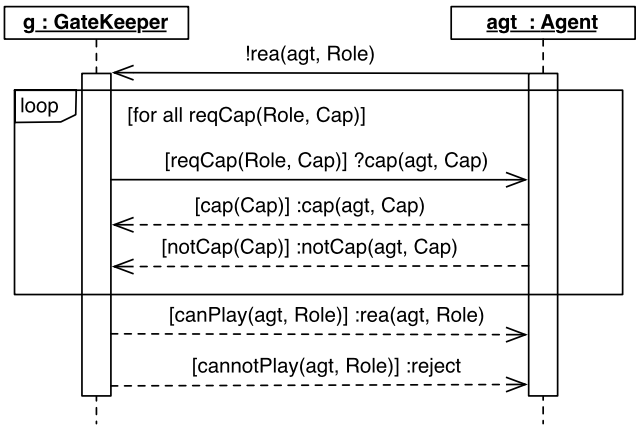


Fig. 2 Role Enactment Interaction Protocol in UML

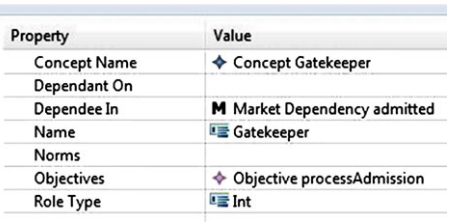


Fig. 3 Standard properties of Gatekeeper

an organisation and the gatekeeper [24]. Figure 2 shows the basic interaction protocol, using the notation of [12] to distinguish different kinds of messages: the prefix "`!`" for imperative messages (requests), "`?`" for interrogative (questions), and "`:`" for declarative (information). The figure shows the interaction where the applying agent `agt` sends a message to the gatekeeper that it wants to play a certain role, i.e., that it wants to become a *role-enacting agent* or *rea* for short [8] (`!rea(agt,Role)`). The gatekeeper replies by asking the agent whether it has the capabilities to play this role (`?cap(agt,Cap)`). It does this for each required capability (`reqCap(Role,Cap)`). The agent replies by informing the gatekeeper of the capabilities it has (`:cap(agt,Cap)` and `:notCap(agt,Cap)`). If the agent has all required capabilities, it can play the role (`canPlay(agt,Role)`) and the gatekeeper informs the agent that it is now playing the role (`:rea(agt,Role)`). If the agent does not have all required capabilities, its request to play the role is rejected.

The enactment protocol can be modified in various ways. In particular, the gatekeeper and the applying agent could negotiate about whether the latter is allowed to play the role even if it does not have all required capabilities. We elaborate on this below.

4.1 Specifying Gatekeeper Behaviour

The behaviour of the gatekeeper is determined by the specification of the Gatekeeper role. Figure 3 shows the basic

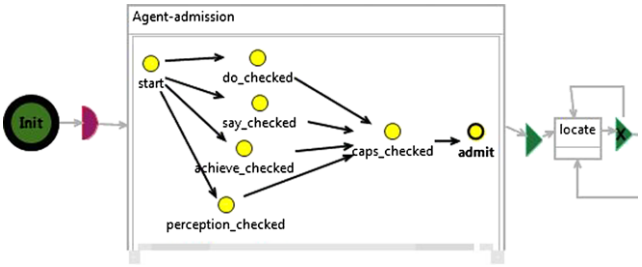


Fig. 4 Interaction Structure for BW4T detailing the Agent-Admission scene

specification of the Gatekeeper role, which is standardly provided by OperA. According to this definition, the gatekeeper can decide to accept any agent that applies for a role. Specifying sub-objectives for *processAdmission* will constrain the gatekeeper to a specific way of admitting agents. For example, sub-objectives

$$request\_capabilities(Agt); check\_capabilities(Agt, Role)$$

$$request\_credentials(Agt)$$

will force the gatekeeper to follow these steps in order to achieve the *processAdmission* objective. A partial definition of the Agent-Admission scene for this situation is depicted in Fig. 4. Landmarks in this scene indicate that the final state *admit* (corresponding to the realisation of the *processAdmission* objective) is entailled from achieving state *caps_checked*, which in turn depends on reaching the states in which each type of capabilities is checked. By refining or relaxing these landmarks, different behaviours can be obtained.

Another way to condition the gatekeeper's behaviour is by defining norms in the role specification. For example, the norm

$$F_{gatekeeper}(admitted(Agt, Role)|(\exists c \in cap(Role) \wedge c \notin cap(Agt)))$$

indicates that the gatekeeper is forbidden to admit agents that do not exactly fulfill all capabilities of the role they aim to play. Other enactment rules can be described similarly. This norm provides a straightforward way of matching the available agents to the role requirements by demanding that each capability of the role can be fulfilled by the agent. That is, if the agent misses even one of the requirements of the role specification, it will not be selected for enactment of that role. This would result in an "all-or-nothing" situation: each role is fulfilled by an agent that has all the capabilities to enact it, and agents with insufficient capabilities are restrained from playing a certain role. In realistic situations, this may be unpractical. In those cases, the norm can be relaxed, for example, to indicate a minimum number of capabilities to fulfill, or to designate some prioritised capabilities that must be fulfilled.

Another possibility would be to keep the norm as above, but to extend the interaction scene to include the possibility for the gatekeeper to suggest a more suitable role to the applying agent, that is, a role for which it does have the required capabilities. For example, in the BW4T scenario, according to this norm an agent without the capability to recognise colour would not be accepted to enact the role of Searcher, but could be suggested to become a Deliverer.

Finally, another possibility for role enactment is to allow one role to be enacted by a set of agents working as a team. In this way, the total capabilities necessary to enact the role should be possessed by the team, which would enable agents to participate that have less capabilities. This is however, an aspect for further research.

4.2 Trust

The Gatekeeper agent has a number of ways to decide on the trustworthiness of the candidate agents. It could check the trustworthiness upfront, or on the fly. To check the trustworthiness upfront, the candidate agents could be required to present a certificate by a trusted third party to the Gatekeeper that states that the agent is indeed capable of perceiving, acting and/or communicating as promised. This is an easy and effective method, but requires such a third party that can actually check the candidate agents. The checking might be done off line. Alternatively, the Gatekeeper can set up a trial run for the agent to see whether the agent indeed has the required capabilities, before admitting the agent into the real team.

Another approach is for the Gatekeeper to consider the reputation of the candidate agent in the community. There is enough agent literature on reputation and trust mechanisms to choose appropriate and effective mechanisms per application. See, for example, [17, 21, 22]. A last and probably least attractive method, is to use a fundamentally much more complex approach in which the Gatekeeper would ask for the essential code of the candidate agent, and then use verification or model checking techniques to test the code for the desired capability. For literature on model checking agents, see e.g., [3, 16].

Instead of, or in addition to, deciding on the trustworthiness and capabilities of the candidate agent the Gatekeeper can also use monitoring methods to form an opinion of the player agents as they perform their tasks. Keeping a memory of results of monitoring of past performance would be used to decide to accept the agent again for the same role. For example, the GRID community has done research on this topic, see e.g., [18, 25]. The level of monitoring can be made dependent on the sensitivity of the application and also depends on the mechanisms put into place to prevent prohibited actions by the agents.

5 Conclusions

If we want to design open organisations, in which heterogeneous agents are able to join, then we must be able to specify what is expected of those agents, and engage in a process of admission during which the requirements and aims of both the organisations and the agent are evaluated. In this paper, we assume that agents are capable of reasoning about role enactment, and focus on the organisation part of this process. In [24] we discussed the reasoning and reflective capabilities of the agents.

We took the simple blocks world scenario as a case study, and used the OperA organisation modelling framework as a basis for the specification of an organisation. For this purpose, we extended the OperA language to include the representation of role capability requirements (i.e. ableToDo, ableToPerceive, ableToSay, and ableToAchieve), which are formally interpreted as norms in the OperA model.

Furthermore, we discussed how role-enactment agreements can be achieved by a process of negotiation between a gatekeeper and a candidate agent. This is specified as an interaction scene in the OperA model, which operationalisation results in the set of role-enactment contracts of the Social Model. We have proposed different behaviours for the gatekeeper which represent possible levels of verification of agent suitability for a role.

In future work we aim to further detail the possible gatekeeper behaviours. Moreover, it would be interesting to investigate the relation between an approach that uses a gatekeeper for matching agents and roles, and work on semantic matchmaking in service-oriented systems. The latter facilitates a more semantic definition of capability and the application of existing flexible matchmaking algorithms. Another direction for future research is investigating the relation between the capabilities required by a role and other norms that agents should adhere to when playing the role. For example, if there is a norm saying that an agent should communicate certain information at a certain point, it would presumably make sense to require the agent to be capable of doing this.

Acknowledgements The authors are grateful to the anonymous reviews for their constructive comments. This works was partially supported by the ESW project (Extended Single Window: Information Gateway to Europe) funded by the Dutch Institute for Advanced Logistics (DINALOG).

References

1. Aldewereld H, Dignum V (2010) OperettA: Organization-oriented development environment. In: Languages, methodologies and development tools for multi-agent systems (LADS2010@Mallow). LNCS, vol 6822, pp 1–19

2. Behrens T, Hindriks K, Dix J (2010) Towards an environment interface standard for agent platforms. Ann Math Artif Intell 1–35. doi:10.1007/s10472-010-9215-9
3. Bordini R, Fisher M, Visser W, Wooldridge M (2006) Verifying multi-agent programs by model checking. Auton Agents Multi-Agent Syst 12:239–256
4. Braubach L, Pokahr A, Lamersdorf W (2006) Extending the capability concept for flexible BDI agent modularization. In: Third international workshop on programming multi-agent systems (ProMAS'05). LNCS, vol 3862. Springer, Berlin, pp 139–155
5. Busetta P, Howden N, Rönnquist R, Hodgson A (2000) Structuring BDI agents in functional clusters. In: ATAL'99: 6th international workshop on intelligent agents VI, agent theories, architectures, and languages (ATAL). Springer, Berlin, pp 277–289
6. Cholvy L, Garion C, Saurel C (2005) Ability in a multi-agent context: a model in the situation calculus. In: Proc CLIMA VI
7. Cross C (1986) 'Can' and the logic of ability. Philos Stud 50:53–64
8. Dignum V (2004) A model for organizational interaction: based on agents, founded in logic. PhD Thesis, SIKS Dissertation Series 2004-1, Utrecht University
9. Dignum V, Dignum F (2012) A logic of agent organizations. Log J IGPL (to appear). doi:10.1093/jigpal/jzr041
10. Hewitt C (1991) Open information systems semantics for distributed artificial intelligence. Artif Intell 47:79–106
11. Hindriks KV (2009) Programming rational agents in GOAL. In: Bordini RH, Dastani M, Dix J, El Fallah Seghrouchni A (eds) Multi-agent programming: languages, tools and applications. Springer, Berlin
12. Hindriks K, van Riemsdijk MB (2010) A computational semantics for communicating rational agents based on mental models. In: Programming multiagent systems, 7th international workshop (ProMAS'09). LNAI, vol 5919. Springer, Berlin, pp 31–48
13. Hübner JF, Sichman JS, Boissier O (2007) Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels. Int J Agent-Oriented Softw Eng 1(3/4):370–395
14. Johnson M, Jonker CM, van Riemsdijk MB, Feltovich PJ, Bradshaw JM (2009) Joint activity testbed: Blocks world for teams (BW4T). In: Proceedings of the tenth international workshop on engineering societies in the agents' world (ESAW'09). LNAI, vol 5881. Springer, Berlin, pp 254–256
15. Johnson M, Bradshaw JM, Feltovitch PJ, Jonker CM, van Riemsdijk MB, Sierhuis M (2010) Coactive design, why interdependence must shape autonomy. In: Proc 9th int workshop on coordination, organization, institutions and norms in multi-agent systems, COIN@AAMAS2010
16. Jongmans SS, Hindriks K, van Riemsdijk MB (2010) Model checking agent programs by using the program interpreter. In: Dix J, Leite J, Governatori G, Jamroga W (eds) Computational logic in multi-agent systems. Lecture notes in computer science, vol 6245. Springer, Berlin, pp 219–237
17. Jøsang A, Ismail R, Boyd C (2007) A survey of trust and reputation systems for online service provision. Decis Support Syst 43(2):618–644. Emerging Issues in Collaborative commerce
18. Lenzini G, Tokmakoff A, Muskens J (2007) Managing trustworthiness in component-based embedded systems. Electron Notes Theor Comput Sci 179:143–155
19. Padgham L, Lambrix P (2005) Formalisations of capabilities for BDI-agents. Auton Agents Multi-Agent Syst 10(3):249–271
20. Penserini L, Dignum V, Staikopoulos A, Aldewereld H, Dignum F (2009) Balancing organizational regulation and agent autonomy: An MDE-based approach. In: ESAW'09: proceedings of the 10th international workshop on engineering societies in the agents world X. Springer, Berlin, pp 197–212
21. Resnick P, Kuwabara K, Zeckhauser R, Friedman E (2000) Reputation systems. Commun ACM 43:45–48
22. Sabater J, Sierra C (2005) Review on computational trust and reputation models. Artif Intell Rev 24:33–60
23. van Riemsdijk MB, Hindriks KV, Jonker CM (2009) Programming organization-aware agents: A research agenda. In: Aldewereld H et al (eds) Engineering societies in the agents' world X (ESAW'09). LNAI, vol 5881. Springer, Berlin, pp 98–112
24. van Riemsdijk MB, Dignum V, Jonker CM, Aldewereld H (2011) Programming role enactment through reflection. In: Boissier O (ed) Proceedings of the 10th IEEE/WIC/ACM international conference on intelligent agent technology (IAT)
25. Yang S, Butt AR, Hu YC, Midkiff SP (2005) Trust but verify: monitoring remotely executing programs for progress and correctness. In: Proceedings of the tenth ACM SIGPLAN symposium on principles and practice of parallel programming, PPoPP'05. ACM, New York, pp 196–205

Huib Aldewereld is researcher at the ICT section of the Delft University of Technology. He got his Ph.D. in 2007 at Utrecht University on the topic of norms and institutions for agent systems. He has since worked at several universities as researcher in the fields of distributed plan diagnosis, organising webservice-based systems, and (agent) organisations for scheduling and planning logistical services.

Virginia Dignum is associate professor in the ICT section at Delft University of Technology. She got her Ph.D. in 2004 from the Utrecht University. In 2006, she was awarded the prestigious Veni grant from NWO (Dutch Organization for Scientific Research) for her work on agent-based organisational frameworks. Her research focuses on agent based models of organizations, in particular in the dynamic aspects of organizations.

Catholijn M. Jonker is full professor of Man-Machine Interaction at the Delft University of Technology. She studied computer science, and did her Ph.D. studies at Utrecht University. From September 2004 until September 2006 she was a full professor of Artificial Intelligence/Cognitive Science at the Radboud University Nijmegen. She chaired De Jonge Akademie of the KNAW (The Royal Netherlands Society of Arts and Sciences) in 2005 and 2006, of which she was a member from 2005 to 2010.

M. Birna van Riemsdijk is assistant professor in the Man-Machine Interaction group at Delft University of Technology. Until September 2008, she was a postdoc at LMU Munich, and she obtained here Ph.D. at Utrecht University. She has done research in the areas of agent programming and service-oriented systems. She is a member of the steering committee of the workshop on Declarative Agent Languages and Technologies (DALT) and has been a co-chair of several workshops.