Reflection about capabilities for role enactment

Conference Paper · January 2011

Source: DBLP

CITATION
1

READS
28

4 authors, including:

Virginia Dignum
Delft University of Technology
312 PUBLICATIONS **3,556** CITATIONS
SEE PROFILE

Catholijn M. Jonker
Delft University of Technology
543 PUBLICATIONS **6,385** CITATIONS
SEE PROFILE

Huib Aldewereld
Hogeschool Utrecht
93 PUBLICATIONS **930** CITATIONS
SEE PROFILE

Reflection about Capabilities for Role Enactment (Extended Abstract)

M. Birna van Riemsdijk[1] Virginia Dignum[1] Catholijn M. Jonker[1] Huib Aldewereld[2]
Delft University of Technology, Delft, The Netherlands[1]
Utrecht University, Utrecht, The Netherlands[2]
{m.b.vanriemsdijk,m.v.dignum,c.m.jonker}@tudelft.nl[1]
huib@cs.uu.nl[2]

ABSTRACT

An organizational modeling language can be used to specify an agent organization in terms of its roles, organizational structure, norms, etc. Using such an organizational specification to organize a multi-agent system should make the agents more effective in attaining their purpose, or prevent certain undesired behavior from occurring. Agents who want to enter and play roles in an organization are expected to understand and reason about the organizational specification. An important aspect that such organization-aware agents should be able to reason about is role enactment. In particular, agents should be able to reflect on whether they have the capabilities to play a role in an organization. In future work it needs to be made precise when an agent can be said to have a certain capability, and how an agent can reflect on its capabilities. This is necessary for programming role enactment in organization-aware agents.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, languages and structures*; F.3.2 [**Logics and Meaning of Programs**]: Semantics of Programming Languages

General Terms

Theory, Languages

Keywords

Organizational Modelling Languages, Organization-Aware Agents

1. INTRODUCTION

An *organizational modeling language* can be used to specify an agent organization in terms of its roles, organizational structure, norms, etc. (see, e.g., [2, 4]). Such an organizational specification abstracts from the individual agents that will eventually play the roles in the organization. Using an organizational specification is a sine qua non for creating open multi-agent organizations that allow agents to join or leave the organization.

Agents who want to enter and play roles in an organization are expected to understand and reason about the organizational specification, if they are to operate effectively and flexibly in the organization. Agents that are capable of such organizational reasoning and decision making are called *organization-aware agents* [6]. Our broader aim is the development of languages and techniques for programming organization-aware agents.

An important aspect that organization-aware agents should be able to reason about is *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. Here we focus on the latter. We are in particular interested in how agents can be programmed to perform such reasoning and take this into account in their decision making about role enactment.

In order to investigate how to program this kind of reasoning, we propose to develop a general pattern for modelling capabilities in the OperA organizational modelling language, in which we distinguish several *capability types*. We propose that agents should be able to *reflect* on their capabilities using their beliefs. These investigations will contribute to the development of languages and techniques for programming organization-aware agents.

Cite as: Reflection about Capabilities for Role Enactment (Extended Abstract), van Riemsdijk, Dignum, Jonker, Aldewereld, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 1231-1232.

2. BLOCKS WORLD FOR TEAMS

The Blocks World For Teams (BW4T) simulated environment [5] has been developed as a testbed for human-agent/robot teamwork. The environment consists of nine rooms that are connected through halls. Colored 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 the specified color sequence. Blocks only become visible once a robot enters the room where these blocks are. Robots cannot see each other. 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. Here we consider agent-only teams since human-agent interaction is not the focus of this paper.

An interface that allows GOAL agents to control the simulated robots has been developed using the Environment Interface Standard (EIS) [1]. Broadly speaking, this standard specifies that agents can control entities in the environment through actions, and agents can observe the environment through percepts that are sent from the environment to the agents. The actions made available to agents are, e.g., `goTo(<Place>)` to move to the specified place (a room, the drop zone or a hall) and `pickUp` to pick up a block (the robot has to be close to the block). Percepts made available to agents are, for example, `at(<Me>,<Place>)` which specifies in which place the robot currently is, and `color(<Block>,<Color>)` which is sent once an agent enters the room where `<Block>` is located.

3. ORGANIZATIONAL SPECIFICATION

The OperA framework [2] proposes an expressive way for defining open organizations distinguishing explicitly between the organizational aims and the agents who act in it. That is, OperA enables the specification of organizational 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.

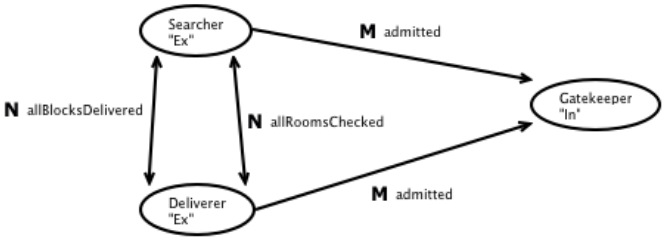
The OperA framework consists of three interrelated models. Here in particular the *Organizational Model* (OM) is relevant. This is the result of the observation and analysis of the domain and describes the desired behaviour of the organization, as determined by the organizational stakeholders in terms of roles, objectives, norms, interactions and ontologies. The design and validation of OperA OMs can be done with the OperettA tool [3]. The OM provides the overall organization design that fulfills the stakeholders requirements.

Figure 1 shows the social structure of the BW4T organization, and the corresponding role descriptions for the $Searcher$ and $Deliverer$ roles. The arcs in the social structure diagram define the dependency relations between the roles. These dependencies indicate how the distribution of objectives in the organisation is realized. The arcs are labelled with the objectives for which the parent role depends on the child role. OperA identifies three types of role dependencies: bidding [**M**arket], request [**N**etwork], and delegation [**H**ierarchy].

In the BW4T example, the organizational objective of collecting the colored blocks in a particular color order is split over the two roles in the organization; the *Searcher*s are responsible for checking all rooms for the blocks and providing the information about block locations and colors to other agents (*allRoomsChecked*), and the *Deliverer*s are responsible for picking up the blocks of the correct color 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 organizational model. The gatekeeper is responsible for admitting agents to the organization 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 organization herself, while the other roles are marked as external ("Ex"). The latter kind of role can be played by agents that are designed independently from the society. Individual agents consider joining an organization 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 realization 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 generated in the Social Model are the result of the these processes.

The normative structure enables the definition of norms that specify desired behavior that agents should exhibit when playing the role. 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. In particular, we propose that norms can be used to express which *capabilities* an agent should have for playing a certain role.



Property	Value
Concept Name	Concept Searcher
Dependant On	N Network Dependency allBlocksDelivered,
Dependee In	N Network Dependency allBlocksDelivered,
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

Figure 1: Role dependencies (top), properties of Searcher (middle) and Deliverer (bottom).

In order to reason about role enactment, agents should be able to reflect on the capabilities that they have. In future work it needs to be made precise when an agent can be said to have a certain capability, and how an agent can reflect on its capabilities. This is necessary for programming role enactment in organization-aware agents.

4. REFERENCES

[1] T. Behrens, K. Hindriks, J. Dix, M. Dastani, R. Bordini, J. Hübner, L. Braubach, and A. Pokahr. An interface for agent-environment interaction. In *Pre-Proceedings of ProMAS'10*, 2010. To appear in LNAI.

[2] V. Dignum. *A Model for Organizational Interaction: based on Agents, founded in Logic*. SIKS Dissertation Series 2004-1. Utrecht University, 2004. PhD Thesis.

[3] V. Dignum and H. Aldewereld. OperettA: Organization-oriented development environment. In *Proceedings of the 3rd International workshop on Languages, Methodologies and Development Tools for Multi-agent Systems (LADS2010@Mallow)*, 2010.

[4] 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 AOSE*, 1(3/4):370–395, 2007.

[5] M. Johnson, C. M. Jonker, M. B. van Riemsdijk, P. J. Feltovich, and J. M. Bradshaw. Joint activity testbed: Blocks world for teams (BW4T). In *Proceedings of ESAW'09*, volume 5881 of *LNAI*, pages 254–256. Springer, 2009.

[6] M. B. van Riemsdijk, K. V. Hindriks, and C. M. Jonker. Programming organization-aware agents: A research agenda. In *Proceedings of ESAW'09*, volume 5881 of *LNAI*, pages 98–112. Springer, 2009.